DOCUMENT RESUME

ED 208 901 IR 010 004

AUTHOR Feurzeig, W.; And Others
TITLE Microcomputers in Education. Report No. 4798.
INSTITUTION Bolt, Beranek and Newman, Inc., Cambridge, Mass.
SPONS AGENCY Ministerio para Desarollo de la Inteligencia Humana, Caracas (Venezuela).; National Inst. of Education (DHEW), Washington, D.C.
PUB DATE Oct 81
CONTRACT 400-80-0031
NOTE 115p.

EDRS PRICE MF01/PC05 Plus Postage.
DESCRIPTORS Artificial Intelligence; *Computer Assisted Instruction; Computer Graphics; *Computer Oriented Programs; *Computer Programs; Cost Effectiveness; Educational Games; Educational History; Educational Research; *Futures (of Society); *Input Output Devices; Man Machine Systems; *Microcomputers; Problem Solving; Programing Languages

ABSTRACT

A brief review of the history of computer-assisted instruction and discussion of the current and potential roles of microcomputers in education introduce this review of the capabilities of state-of-the-art microcomputers and currently available software for them, and some speculations about future trends and developments. A survey of current applications of microcomputers is followed by descriptions of hardware capabilities, including input and output devices and displays, as well as memory, and a review of some representative software for teaching mathematics, language skills, social sciences, natural sciences, and educational games. The status of research on the instructional effectiveness of CAI is briefly discussed, and five prominent studies which show consistent results are cited. Descriptions of near-future technological capabilities include hardware trends, software developments (including high level languages, operating systems, and user interface), artificial intelligence, and programming languages. Changing expectations, potential problems, and educational promise are addressed in a concluding discussion of prospects and pitfalls. A 43-item bibliography is provided. (CHC)

****
* Reproductions supplied by EDRS are the best that can be made *
* from the original document. *
****

ED208901

U.S. DEPARTMENT OF EDUCATION
NATIONAL INSTITUTE OF EDUCATION
EDUCATIONAL RESOURCES INFORMATION
CENTER (ERIC)
☒ This document has been reproduced as received from the person or organization originating it.
☐ Minor changes have been made to improve reproduction quality.
• Points of view or opinions stated in this document do not necessarily represent official NIE position or policy.

Report No. 4798

MICROCOMPUTERS IN EDUCATION

W. Feurzeig, P. Horwitz, and R. S. Nickerson

October 1981

Prepared for:

Department of Health, Education and Welfare
National Institute of Education
and
Ministry for the Development of Human Intelligence
Republic of Venezuela

Preparation of this report was supported by the U.S. National Institute of Education under Contract No. 400-80-0031 and by the Republic of Venezuela under Purchase Order No. M76075.

IR010004

TABLE OF CONTENTS

		Page
1.	INTRODUCTION	1
1.1	Background	1
1.2	Purpose of Paper	4
1.3	Why the Interest in CAI?	4
1.4	Content and Organization of Paper	18
2.	REVIEW OF CAI METHODS	21
2.1	Computer-Controlled CAI	21
2.2	Student-Controlled CAI	23
2.3	Mixed Initiative Systems	25
2.4	Computer Languages as Learning Environments	31
3.	SURVEY OF CURRENT EDUCATIONAL APPLICATIONS OF MICROCOMPUTERS	33
3.1	Prescriptive CAI	33
3.2	Educational Simulations and Games	35
3.3	Educational Production Aids	36
3.4	Educational Computer Languages	38
3.5	Some Observations Regarding Usage	42
4.	CURRENT MICROCOMPUTER HARDWARE AND SOFTWARE	47
4.1	Hardware Capabilities	47
4.1.1	Computing element (CPU) and primary memory	48
4.1.2	Secondary storage	49
4.1.3	Screen output	49

4.1.4 Printed output 50
4.1.5 Input devices 51
4.1.6 Sound output 52
4.1.7 Speech input and output. 52
4.2 Representative Software 53
4.2.1 Mathematics 53
4.2.2 Language Skills 58
4.2.3 Social Sciences 59
4.2.4 Natural Sciences 64
4.2.5 Educational Games 68

5. EVALUATIONS OF EDUCATIONAL EFFECTIVENESS 73

6. NEAR-FUTURE TECHNOLOGICAL CAPABILITIES 77

6.1 Hardware Trends 77
6.1.1 The Central Processing Unit 78
6.1.2 The Memory 80
6.1.3 Input Devices 81
6.1.4 Output Devices 82
6.2 Software Developments 84
6.2.1 High Level Languages 85
6.2.2 Operating Systems 86
6.2.3 User Interface 87
6.3 Artificial Intelligence 89
6.4 Programming Languages 92

7. PROSPECTS AND PITFALLS 97

7.1 Changing Expectations 97
7.2 Potential Problems 98
7.3 Educational Promise 106

ABSTRACT

Computer assisted instruction has been a topic of considerable interest among researchers and educators for more than 20 years. To date, accomplishments have not matched early expectations. One of the major limitations to progress in this field has been the high cost of computing resources. Recent advances in microelectronics technology and, in particular, the development of the computer on a chip, have greatly reduced the costs of computing hardware, and have also made possible the packaging of very large amounts of computing power in portable devices. The trend of providing increasing amounts of computer power in increasingly smaller packages and at ever decreasing cost is expected to continue into the foreseeable future. This report is motivated by the belief that this trend has very substantial implications for the future use of computer technology in education. Following a cursory review of the history of computer assisted instruction, the capabilities of state-of-the-art microcomputers are described. Currently available microcomputer software with potential educational uses is then surveyed; and some anticipated future developments are discussed.

1. INTRODUCTION

1.1 Background

There appears to be a growing interest in the explicit teaching of thinking skills in the classroom. Efforts have been, and are being, made in several countries to develop and implement programs to enhance the ability of students to think creatively and critically, and to perform more effectively on a broad spectrum of intellectually demanding tasks. (For a review of several of these programs see Nickerson, Perkins and Smith, 1981.)

A common characteristic of most if not all of these programs is that they are neither individualized nor adaptive. An entire class works on the same material at any given time and is expected to proceed through the course at the same rate. Programs to teach thinking skills are not unique in this regard. Probably the vast majority of courses taught in formal school settings are neither individualized nor very adaptive.

These comments are not intended to be critical of traditional approaches to classroom teaching. Paradoxically, the clearly desirable policy of universal free primary and secondary education has made it necessary to use "mass" teaching techniques. But everything we know about human learning makes it

clear that the mass approach must be inefficient at best. People do not learn at the same rates, and the same path through a given set of materials is not likely to be equally effective for all learners. Moreover, any given person will be more or less attentive at different times, and therefore will not be able always to perform optimally when the pacing of material is not under his control.

This is not really a controversial issue. Researchers and educators alike have acknowledged the desirability of individualized and adaptive instruction--instruction tailored to meet the individual's needs and continually modified so as to be appropriate to his current level of understanding and performance. While the desirability of individualized adaptive instruction has been widely recognized, the approach has not been a feasible one, except in a few unusual situations, because of the prohibitive cost of implementing it. Individualization requires a very low (ideally, one-to-one) student to teacher ratio. Adaptability requires great flexibility on the part of the teacher, and precludes the tightly prescribed single-path approach to lesson preparation that is so widely followed.

Since people first began thinking about the use of computers in education, there has been some hope that computer-assisted instruction would represent an answer to this problem and would establish the cost-effectiveness of individualized adaptive

teaching. To date, this hope has not been realized. Indeed, what has been accomplished has fallen so far short of many of the early predictions, that hope has given way to skepticism, and in some circles any attention that is given to computer-assisted instruction is viewed as an unfortunate diversion from the serious business of developing more effective educational techniques.

We believe that computer-assisted instruction (CAI) does, in fact, hold great potential for the development and use of individualized adaptive educational techniques. Without doubt, many of the early predictions regarding the future of this technology were overly optimistic with respect to how quickly computer-based techniques could be made both educationally effective and financially competitive with other approaches. The disillusionment that has occurred is a result not of having discovered that the technology does not have the potential it was once thought to have; rather, it is a consequence of having been forced to realize that the effective application of that technology to educational purposes is a much more difficult task than it was once thought to be. Unfortunately, the two issues have sometimes been confused, and people have interpreted the fact that the promise has been delayed as evidence that it will never be realized.

1.2 Purpose of Paper

The purpose of this paper is to attempt to reassess the prospects for cost-effective and educationally-effective CAI in the foreseeable future, in light of what has happened in CAI in the past and of recent developments in computer technology, particularly the increasing availability of very powerful microcomputers. Teaching techniques that are individualized and adaptive in any but the most trivial of senses requires the programming of complex procedures and the ability to store a large knowledge base. In the past, these requirements have necessitated the use of large and very expensive machines. Computer technology has made phenomenal progress in recent years, however, in packaging larger and larger amounts of computing power in smaller and smaller units, and at steadily decreasing costs. If one extrapolates recent trends, one must conclude that the time is not far off when the factor that will limit the usefulness of computer technology in education is not the cost of the hardware, but our ability to invent ways of using it effectively; indeed that time may already be here.

1.3 Why the Interest in CAI?

It is interesting to note that many of the programs that have been developed to teach thinking skills place heavy emphasis on exploration and discovery. In particular, approaches that

stress the engagement of students in investigatory and creative activities are preferred to lecturing. A suitably programmed computer can provide unusual opportunities for exploration and discovery. It can, in fact, constitute a new species of learning laboratory. It is worth illustrating the point with a few examples.

Consider a program that will display on demand a specified mathematical function, say a polynomial of nth degree. Suppose further the program is able, in response to simple commands, to perform such operations on the function as differentiate it, integrate it, add roots, delete roots, multiply by a constant, and so on, and to show how each operation affects the graph of the function immediately after it has been performed.

Or consider a statistics program that permits a student to do simulated experiments with random variables, and watch the operation and consequences of specified stochastic processes. Or a program that permits one to choose the parameters of a dynamic model of a complex system such as an economy, and observe the effects on a variety of dependent variables of interest. Or a program that permits one to write an n-part musical composition, specifying the instruments for the various parts, and then immediately to hear it played, while simultaneously seeing the score displayed.

Or consider the possibility of students using a programming language as a creative tool for designing and producing mathematical and artistic drawings. Such drawings can, in fact, be made by a computer-controlled device, the LOGO Turtle, associated with the LOGO educational programming language. The turtle's motions and writing are directed by LOGO drawing procedures written by the student. Figure 1 shows a number of drawings made this way by LOGO programs written on the Apple II microcomputer (Abelson, 1981).

The sequence of four LOGO drawings on the left side of the figure shows three distinctly different complex patterns, each of which involves replication of the same motif, provided by the figure shown at the top left. This figure, called SHAPE, is produced by a LOGO procedure consisting of a sequence of drawing commands -- alternating line segments and right angle turns. The figure below SHAPE, called SHAPE4, is produced by a LOGO procedure that merely repeats execution of the SHAPE figure four times. The next figure down, CRYSTAL, is produced by drawing SHAPE, performing a 45 degree left turn, drawing a straight line segment, and then repeating this sequence. The bottom figure on the left, JENGU, is produced by drawing SHAPE twice, performing a 90 degree left turn, and then repeating this sequence. The four figures on the right side of the page are various patterns produced by comparably straightforward executions involving basic circle drawing procedures.

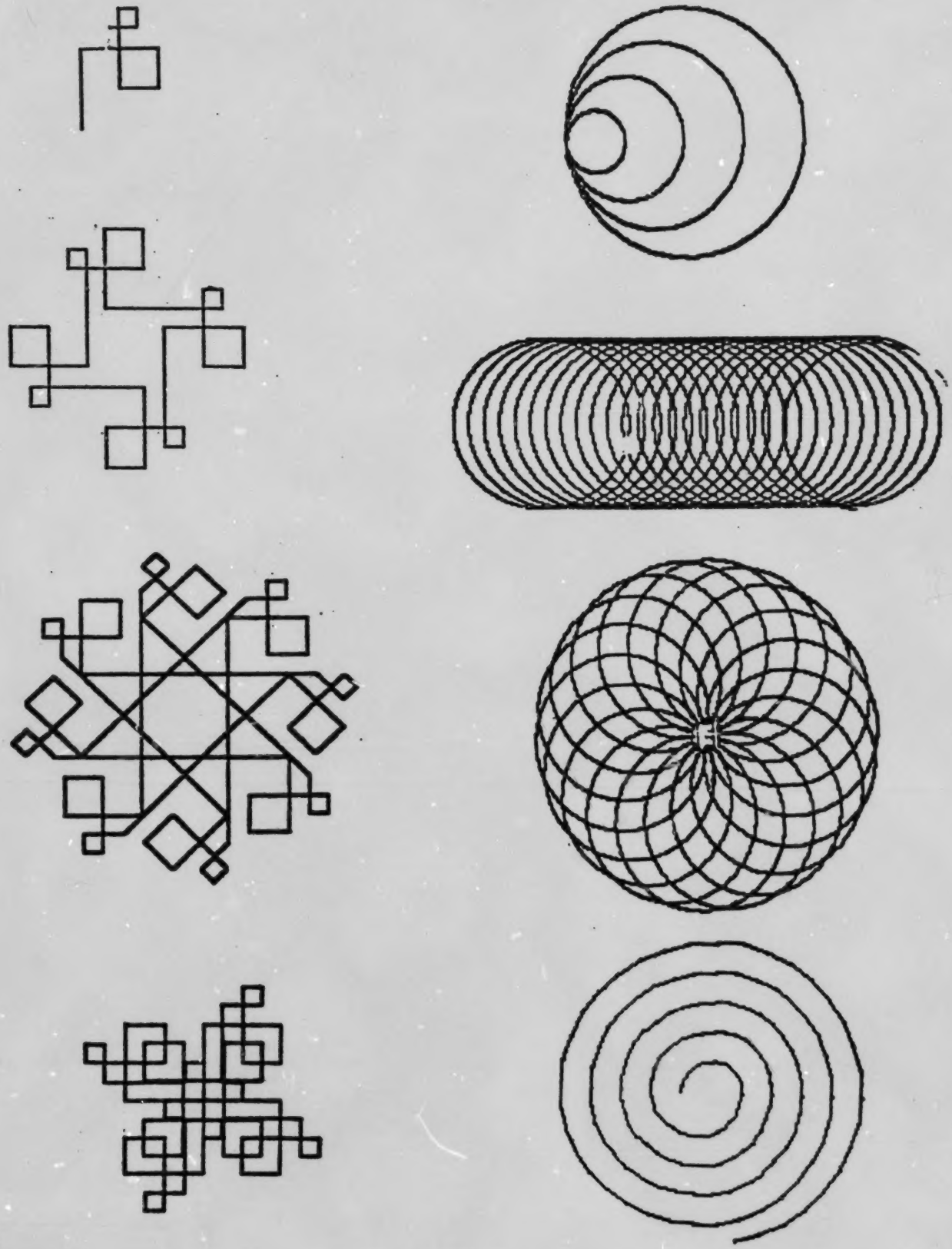
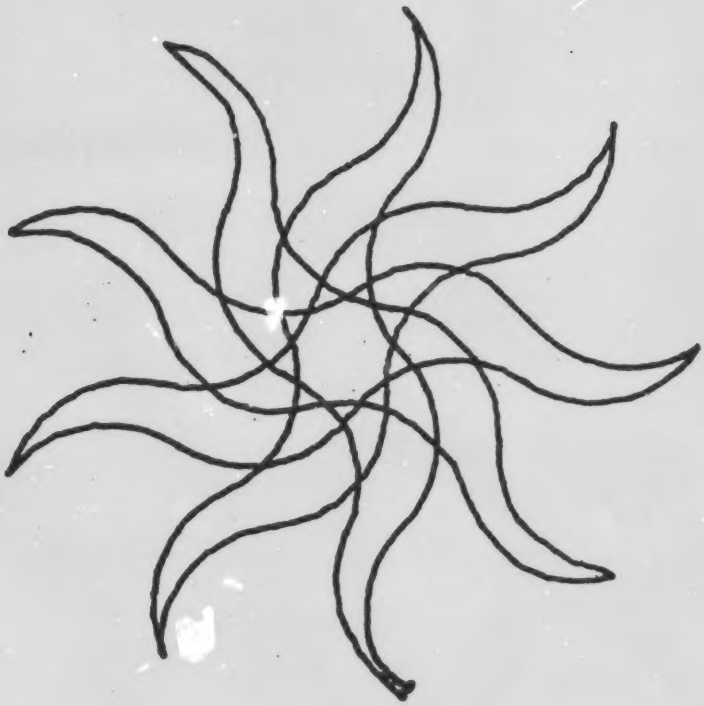


Figure 1. LOGO Turtle Drawings

Figure 2 illustrates how relatively complex drawing procedures are developed out of sub-procedure building blocks. The figure at the top of the page, called SUN, is produced by the three-line LOGO procedure shown. This procedure performs the subprocedure WIGGLE (shown in the middle of the page), executes a 180 degree right turn, and then repeats this sequence (by invoking itself recursively). The procedure WIGGLE, in turn, invokes two subprocedures RCP and LCP, repeating each ten times. The short circular arcs produced by performing these basic procedures RCP and LCP (standing for right circle piece and left circle piece, respectively) are shown at the bottom of the page.

Drawing projects of these kinds are very compelling to children. The activities involved in these developments are relevant not only for teaching the specific concepts and methods of design, drafting, computation, and geometry but also for teaching thinking and problem-solving skills generally. Programming work in this domain provides children highly motivating tasks. In this work the concepts of sub-problem and sub-goal arise naturally and sharply. The work provides a concrete sense of formal methods and their purpose. It provides models both for the global planning of an attack on a problem and the formal details of an elaborated solution. It helps students to organize their approach to formulating and solving certain types of problems, and, hopefully, to develop systematic and rigorous methods of thought more generally.

```
TO SUN  :SIZE
REPEAT 2 [WIGGLE :SIZE]
RIGHT 180
SUN  :SIZE
END
```



```
TO WIGGLE  :SIZE
REPEAT 10 [RCP :SIZE]
REPEAT 10 [LCP :SIZE]
END
```

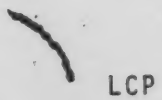


Figure 2. SUN Drawing and Drawing Procedures

In addition to the fact that suitably programmed computers can be used to teach many of the facts, principles and procedures that traditionally have been taught in other ways, there are other arguments for bringing computers into the classroom and teaching students how to use them. Among the more obvious reasons for exposing students to computers during primary and secondary school experiences is the increasing importance of computer literacy for the future. Inasmuch as computers will play ever greater roles in the making of decisions that affect our lives, it will become increasingly important for the average person to have some understanding of what computers can and cannot be expected to do. Many of today's adults have very strange ideas indeed regarding the capabilities of computers. An inaccurate model of the machine's capabilities and limitations can give rise to unreasonable expectations. It can lead to unfounded and extreme attitudes ranging from inordinate fear of the presumed omniscience of these machines on the one hand to unquestioning faith in the same presumed omniscience on the other. Can there be a better way of assuring reasonable and realistic attitudes toward computers than that of giving students a basic understanding of how they work and of the nature of the programs that govern their behavior?

One argument in favor of teaching primary and secondary students how to program is that the ability to program is an

immensely practical one. The opportunities to apply this ability to advantage in post-secondary education and in the workplace outside of school are increasing steadily and rapidly. There is another aspect of the experience of learning to program that is more germane, however, to the topic of teaching thinking skills. That is the possibility that the experience could facilitate the development and improvement of those skills in several ways.

Successful programming requires that one be able to conceive and express procedures for accomplishing specific tasks. It forces one to use language precisely, to make one's assumptions specific, to anticipate exceptions to general rules, to design explicit approaches to problems, to generate and test hypotheses (in diagnosing and finding program bugs), to recognize the difference between effectiveness and efficiency, and so on. The usefulness of each of these abilities extends far beyond the domain of computer programming. Whether the ability to do these things in the process of writing programs transfers to other contexts is an empirical question and one that deserves some study. Our guess is that if the transfer is not spontaneous, it probably could be facilitated if adequate attention were given to that goal.

Experience in programming also helps students discover that typically there is no best way to do something. Usually there are many ways, some of which are better than others, and most of

which can be improved upon. Students also discover that it is sometimes not easy to decide which of several possible procedures is the best among the available alternatives. Often one procedure is better in some respects and another better in others. What determines which will be preferred probably depends on the specifics of the situation in which it is to be used. These are important lessons that also extend well beyond the domain of programming and computer use.

Other arguments for using computers in the classroom include the following:

- The possibility of greater student control over the selection and pacing of material: Evidence seems to show that people learn more effectively when they participate in the selection of what they are learning than when they do not.
- The feasibility of immediate feedback: Another advantage of using a computer in learning situations is the immediacy of the feedback that it provides. One sees the effect of one's work as soon as the work is done. This means that one has the opportunity of using "errors" and "unsuccessful" performance as occasions for learning rather than simply as evidences of failure.
- The motivational benefits of a responsive machine: A particularly motivating aspect of the use of computers for educational purposes is the reinforcement the student gets from actually making something happen. When an executable program has been written, one can in fact execute it and observe the results. To young people who have not often had the experience of having anything under their control, this can provide a real sense of accomplishment.

In keeping with the topic of this paper, the foregoing

comments have focused on uses of computers for explicitly educational purposes. Any attempt to assess the educational potential of computers in a broad sense must also consider, however, the various recreational purposes for which they are being, or will be, used. Inasmuch as people can learn and be entertained at the same time, the line between educational and recreational uses of computers should not be sharply drawn. Indeed, we believe the computer offers an unusual opportunity to accomplish educational objectives by recreational means. Consider, for example, the idea of a dynamic crossword puzzle. Crossword puzzles are a major form of entertainment. In the United States alone an estimated 30 million people do crossword puzzles on a fairly regular basis. Presumably very few people do them primarily for the purpose of increasing their vocabulary; they do them because they enjoy doing them. One suspects that the vocabularies of crossword puzzle doers do benefit, however, from this activity. However, they probably do not benefit nearly as much as they might, because puzzle doers often fill in only the words they know and do not bother to look up the answers to those they missed.

One can imagine a dynamic crossword puzzle that would operate in the following way: The computer displays a crossword puzzle and gives the user as many clues to each word as he wants, one at a time. More specifically, the user is always free to

request a clue for any word--e.g., 7 across, 16 down. When he requests a clue for a word for which he has already had one or more clues, the computer simply provides the next one in the sequence for that word. The clues for a given word become increasingly revealing until, after a certain number of clues have been given, the computer provides the word. Whenever the user wishes to do so, he enters a word that he believes to be appropriate into the puzzle. If it is appropriate, the word remains in the correct location in the puzzle; if it is not appropriate the computer rejects the word. The object of the game is to fill in the puzzle using the smallest possible number of clues. What makes such a computer-based crossword puzzle different from the more conventional type found in the newspaper is that, unless the user quits, the puzzle always gets completely filled in, and hopefully the puzzle doer gets more new information about words--especially words with which he is least familiar--than in the conventional case.

Another good example of a program written for recreational purposes that may have educational benefit is the fantasy simulation program. While placing the player in an imaginary world filled with trolls and dragons, magic spells and bags of gold, such a program may also sharpen language skills and problem solving abilities. An example of one such program may suffice to make the point.

In playing one of the most popular of the fantasy simulation games, you are placed by the computer in the Great Underground Empire of Zork. As you progress through this world, the computer prints out messages describing your immediate surroundings. It also accepts a variety of typed commands that enable you to move about, pick up objects, and defend yourself against a myriad of dangers. Soon after your introduction to this imaginary world, you may find yourself in a small house consisting of an attic, a kitchen, and a living room. Having thoroughly investigated the house and examined carefully every object in it, you are at a loss as to how to proceed. The description of the living room has noted the fact that it has a rug on the floor. After some minutes' thought, you suddenly decide to look underneath this rug. The computer prints out "With great difficulty you move the rug to one side, revealing a trap door". You have just solved one of hundreds of puzzles posed by this "game".

Although the program from which this example is taken is not meant to be used as an educational tool, its ability not only to intrigue and involve children with a fantasy world that exists only in their imagination, but to challenge them with problems of widely varying complexity (from defending oneself against a troll with a sword to draining a dam and navigating the resulting body of water in an inflatable "magic boat"). The interactive environment and the degree of randomness available using the

computer, make simulations, whether of real or imaginary worlds, valuable teaching tools.

For a final example of the potential of recreational programs for education, consider the problem of teaching physics. One of the most difficult concepts to communicate to the beginning student is that embodied in Newton's First Law of Motion: "Every body continues in its state of rest or uniform motion in a straight line, unless acted upon by an external force". Deceptively simple when stated so baldly, this fundamental natural law was not discovered until comparatively recently, precisely because it is both subtle and counter-intuitive. The Aristotelian model which it replaced, and which holds that all motion inevitably slows and eventually ceases, is much more consonant with "commonsense".

The reason for the discrepancy between the two descriptions of reality is summed up in one word: friction. The frictional forces that inevitably act to decrease the relative motion between the parts of a system that are in contact are not easily observable and are neglected in most practical situations. The physics teacher is thus faced with the paradoxical situation that the fundamentally simple, ideal frictionless world of Newtonian mechanics is initially strange and forbidding in comparison to its more complicated, but more familiar, Aristotelian counterpart.

How can the computer help? Two popular electronic arcade games illustrate how computers may provide an answer. In one, called Speedway, the player is placed in the position of driving a car at high speed along a curving track, in the other, Asteroids, he is in command of a rocket moving through (frictionless) space, evading or exploding a variety of objects that come at him from all directions. In Speedway, the mechanics are (barring the inevitable "spinouts" that occur at high speeds) Aristotelian: The velocity vector always lies along the axis of the automobile. The Asteroids game, on the other hand, is governed by Newtonian laws. The player quickly learns that here the velocity vectors add and there is no frictional force to slow his forward momentum during turns. In contrast to the car, the rocket is under no compulsion to move only in the direction in which it is pointed.

Would not these two contrasting games make a marvelous teaching tool? In Speedway the frictional force "required" to keep the car on the road during all maneuvers could be explicitly displayed. In addition, the coefficient of friction could be varied, either automatically or under student control, to demonstrate the evolution from an Aristotelian to a Newtonian world, and the reason for the differences between them. In Asteroids, a continuously updated display of the velocity and acceleration vectors could be used to generate an intuitive feel

for vector addition, which is a notoriously difficult concept for beginning students. At a more advanced level, with the addition of a central gravitational force the same "game" could be used to teach the fundamentals of orbital mechanics.

Can one doubt the educational potential of such tools? By themselves, such facilities would permit exploration and discovery by which many students would surely be able to learn much. In the hands of a competent teacher, who could gently guide the exploration in directions that would maximize the chances of discovering important principles and relationships, such tools could be powerful indeed.

It is easy to imagine many recreational activities that could be programmed for computers that would unquestionably have educational value. There are indeed many recreational programs already in existence that teach as well as entertain. To be sure, what exists does not come close to exhausting the possibilities, even of current computer technology, but it is a beginning.

1.4 Content and Organization of Paper

In what follows we review, in a cursory way, the history of computer-assisted instruction by way of providing a frame of reference in terms of which to discuss the role--current and

potential--of microcomputers in education. We then describe the capabilities of state-of-the-art microcomputers, and survey the software that currently exists for them.

The survey is not exhaustive. It is fairly extensive, however, and we believe it is representative in its coverage of the type of software that is currently available. Comments of an evaluative nature are based largely on opinion. This is so of necessity, inasmuch as evaluation data are very scarce.

Following the description of existing software for microcomputers we speculate a bit about future trends and anticipated developments. In looking ahead it is necessary, of course, to distinguish between the possible and the practical. In the case of CAI, what is practical can be limited either by the state of the art of computer technology, or by our limited ability to apply that technology effectively to educational purposes, or by both. Whereas the state of the art of computer technology has been a major limitation in the past, we believe that our limited ability to apply this technology effectively is rapidly becoming the predominant factor and will increase in significance in the future. There has been, and is, a great disparity between the potential and the actual. We suspect that although much will be accomplished in the foreseeable future, that disparity will increase.

2. REVIEW OF CAI METHODS

In this chapter we review in a general way various CAI techniques that have been developed. Although the paper is focused primarily on the use of microcomputers in education, in reviewing CAI techniques, we consider techniques that have been used with minicomputers and "full size" computers or "mainframes" as well. We do this for two reasons: (a) CAI has been around much longer than microcomputers and most of the techniques in use were developed on larger machines, and (b) programs that run only on mainframes now will be runable on microcomputers in the near future. In short, although the classification of techniques that we find useful is derived from a consideration of the ways in which minicomputers and mainframes have been used, it is applicable to microcomputers as well.

2.1 Computer-Controlled CAI

One branch of CAI views learning as received knowledge. This view derives from early work in programmed instruction and teaching machines. In programs written with this view as a starting point, the computer typically presents a frame containing a portion of the lesson material to be learned, asks a question, evaluates the student's response, selects the next lesson frame based on that response, and, optionally, keeps a running cumulative score. The student's role is purely

responsive. He can only answer the questions that are put to him; he cannot ask questions of the system. (Answer evaluation is usually denoted response analysis by the designers of these computer-controlled CAI applications.) This kind of CAI has been characterized as a "hospital" model of education in which the student is regarded as a patient to be treated (or "programmed") by the computer.

Computer-controlled CAI is promoted as providing individual tutoring but the sense of "individualization" is simply that one student may receive a different sequence of prestored frames than another. The student's frame sequence, however, is not created dynamically to match an ongoing evaluation of his needs. Also, the sense of "tutoring" is merely that there is a one-on-one interaction. The program does not build a model of the student's knowledge and conceptual gaps, and it does not diagnose his difficulties along the way.

Computer-controlled CAI applications are useful for teaching the acquisition of facts and for helping students memorize and consolidate factual knowledge. The two major varieties of computer-controlled CAI are drill-and-practice and tutorial CAI. In drill-and-practice work there is often no exposition, merely the presentation of a question or problem, followed by evaluation of the answer. Tutorial lessons, in contrast, often incorporate extensive branching sequences.

There are a number of current languages and systems for developing and administering computer-controlled CAI. These include COURSEWRITER, PILOT, PLANIT, ASET, TUTOR, CAN and NATAL. Extensive computer-controlled CAI lesson materials have been developed for virtually all school subjects at all levels. Some applications permit the student to choose the instructional unit or lesson to be studied as well as the level of difficulty and mode of presentation of the material. Some systems that have these selection facilities are called "learner-controlled" CAI systems but the sense in which they give the learner control of the instruction is quite limited.

2.2 Student-Controlled CAI

In another major line of CAI development, student controlled CAI, learning is viewed as an active process of investigation. The student is thus given significantly greater scope for initiative-taking and control than in computer-controlled CAI.

Programs that provide the student with complete control over the student-computer interaction are atypical but they do exist. One such program, POLYNOM, displays a graph of any polynomial function the student specifies. The same program permits the student to differentiate, integrate, or perform other operations on the function and immediately see the effects on the graph. Another program permits the student to display a statistical

distribution (e.g., a binomial) and to see how the shape of the distribution changes when the student changes its parameters. Still others simulate certain physical or statistical processes and permit the student to investigate what happens when the values of the variables that underlie the processes are changed.

In contrast with computer controlled CAI, programs of these types exercise no control over the learning situation at all. The computer simply responds within the limitations of the program's repertoire, to the student's commands. A reasonable way to characterize these programs is to say that they provide to the student a sort of laboratory with which he can conduct "try and see" experiments. They do not prescribe what experiments the students should perform nor do they coach the student in any way. They simply follow the instructions the student provides. Such programs can nevertheless be very powerful instruments for exploratory learning. If, for example, one knows a little about polynomial functions and has a program that can draw the graphs of such functions instantaneously and also perform the various operations that one typically wants to perform on these functions, one can, by experimenting with such a program, increase one's knowledge very considerably in a relatively short time. Similarly a program that simulates certain aspects of a physics laboratory and permits a student to define simulated physics experiments can be a powerful tool for facilitating the

acquisition of certain fundamental concepts and relationships in physics.

2.3 Mixed Initiative Systems

Mixed initiative systems come much closer than either the computer controlled or student controlled systems to capturing the give and take that one associates with a Socratic approach to teaching and learning. Student and computer engage in a dialog in which the student can ask questions of the computer and the computer is able not only to respond to the student but to ask its own questions in turn.

Typically with mixed initiative systems, the student works on a complex problem-solving task. Examples are developing proofs in logic or geometry, analyzing unknown chemical substances, and finding faults in electronic circuits. The problem may have multiple solutions or no solution-- the emphasis is on learning problem-solving procedures and strategies rather than on finding "the answer." Ideally, the student is given substantial freedom to direct his own investigation within the constraints imposed by a prespecified universe of discourse. He can ask questions as readily as the computer; thus the term mixed-initiative CAI is sometimes used to describe this kind of instructional interaction.

As in the case in student-controlled CAI, the precise actions taken by the student, and their sequencing, are not predictable in advance. In order to serve effectively as guide and critic to the student, the system must have considerable knowledge about the problem or problem domain (this is called a "knowledge base"), as well as capabilities for responding to the student's inputs in a context-sensitive and history-dependent fashion. In the course of the interaction, the system may engage the student with comments and questions along lines such as the following.

Here is a problem to explore, resolve, or solve. Can you solve it? How do you proceed? What do you know to start with? What do you need to know? Why? How can you find out? Try. Ask me; I may have information that would help. Why do you want to know this? How will it help you? What will it cost to do this? Did it help? Did it reduce or expand the range of possibilities? What else do you need? Do you have any hypotheses? Is there a pattern? Is that consistent with other information? How would you test it? Tell me when you have reached a decision. Are other conclusions possible? How could you decide? How do you know it isn't X? That's not consistent with Y, is it? Had you done X, you would know Y. What action will you take? Are you sure?

To carry out an instructional dialogue of this kind in the

sustained and deep manner implied, the system must have powerful instructional strategies. Systems with these sophisticated capabilities can be realized in two ways -- either by developing extensive scenarios that explicitly describe all response contingencies in advance or by developing "expert" programs that are capable of solving the problems they teach.

The former approach is used in the MENTOR mixed-initiative CAI system (Feurzeig, 1969). MENTOR was designed as a conceptual framework for teaching problem-solving strategies and methods. The student is engaged in an investigative task. His actions and the associated responses of the system (which may be highly dependent on history and context) are described in a, possibly extensive, scenario. The MENTOR system includes a programming language for authoring such scenarios. In MENTOR, strategies are expressed by conditional expressions of the form: situation --> actions. Given a certain "situation" the corresponding "actions" are performed. Characteristic types of situations specify a particular student input and knowledge state representing, for example, prior information elicited by the student. Actions include messages answering student questions or questioning student decisions and system interventions of various kinds. MENTOR has been used to teach problem-solving in areas including medical diagnosis, electronic troubleshooting, and management decision-making (Feurzeig, Goldenberg, Mattlage, Stachel, 1979).

Instructional systems using artificial intelligence (AI) methods are sometimes called "intelligent" CAI (ICAI) systems. These systems are designed to behave like experts in specific task domains. ICAI systems have capabilities for interpreting highly varied student responses, diagnosing "bugs" in a student's reasoning, explaining correct solutions, and giving the student advice about what to do next.

One of the earliest ICAI systems, SCHOLAR, (Carbonell, 1970) is capable of sustaining an extended dialogue with a student on South American geography, using its substantial knowledge of the physical, economic, social and cultural aspects of South America to answer a student's questions and to pose related questions to the student. Another system (Goldberg, 1973), developed at Stanford University, is a theorem-proving program capable of checking student proofs in logic. A more recent system, BUGGY, is capable of automatically synthesizing a deep-structure model of students' errors in solving arithmetic problems (Brown & Burton, 1978). The program teaches diagnosis of the underlying faulty procedure accounting for the student's errors. Other sophisticated instructional systems based on artificial intelligence techniques have recently been developed for teaching medical diagnosis (Clancey, 1979).

One of the most elaborate ICAI programs developed thus far is the SOPHIE system designed to tutor students in

troubleshooting an electronic circuit (Brown & Burton, 1974). SOPHIE employs, as part of its knowledge base, a complex circuit simulation program to evaluate the performance of an electronic circuit under various conditions.

In a typical instructional session with SOPHIE, a student is presented with a problem involving a faulty power supply. The student tries to determine the fault by requesting measurements. At any time, he can offer a hypothesis about what he thinks could be wrong; and SOPHIE will evaluate this hypothesis in terms of the measurements made. The student can request that a component be replaced at any time. Before the part is actually replaced, the student is required to say exactly what he thinks is wrong with it. If the replaced part is actually a "secondary fault," the replaced component will be blown again. As in the repair of real instruments, the fundamental underlying fault must be diagnosed and repaired first.

A major component of the system, called the Expert Debugger, can locate faults in the instrument, and articulate the deductions that led to the diagnosis. It instructs by first letting the student choose a fault and then engaging him in a dialogue as it progresses insightfully toward finding the fault, explaining its strategies along the way.

In MENTOR and in ICAI systems, the program develops a

dynamic model of the student's knowledge and uses this to respond to the student, and possibly to intervene, in a history-dependent fashion, taking into account the student's state of knowledge and the actions he has taken thus far in the interaction. Some systems use the model to try to infer the student's strategies, conceptual gaps, and "mind bugs." Some systems have an explicit "coaching" component that, based on such information, tries to advise and guide the student throughout the course of the interaction. Examples of such coaching systems include the program WUMPUS, which is designed to teach problem-solving strategies in a logic game, and an ICAI version of the arithmetic teaching game "How the West Was Won", which can advise the student concerning his computational strategies (Resnick, 1975).

ICAI programs are large and complex. The development of such programs involves substantial research. Their implementation calls for an extensive programming effort, and their operation requires relatively large and powerful computers. For these reasons, ICAI programs will probably not be widely available on first-generation educational microcomputers. However, inexpensive computer systems with the required capabilities will very probably be available during the 1980's.

2.4 Computer Languages as Learning Environments

Another approach to the use of computers in education is orthogonal to the CAI approaches described above in that the computer does not employ an explicit teaching program. In this kind of approach, the student uses a computer language as a learning environment. As with the student- controlled CAI systems, the student commands the computer, rather than the reverse. Here, however, the student is given the enormous expressive power of a universal high level programming language. The range of problems and tasks accessible to programming is, of course, very large. The nature of the learning experience is substantially different from those provided through purely prescriptive CAI methods. Some educational benefits arising from the use of student programming languages are described in Section 3.4.

3. SURVEY OF CURRENT EDUCATIONAL APPLICATIONS OF MICROCOMPUTERS

As a means of organizing the necessarily large amount of material covered by this survey, it is useful to keep in mind that there is a continuous spectrum of computer applications (as discussed above), ranging from the use of the computer purely as a prescriptive teaching device, to applications in which it is considered a tool of the student and is used to further the student's own purposes. In this section we shall give generic examples of program categories that fall along this spectrum, and report some data regarding current uses of microcomputers in education. In subsequent sections we shall describe more fully specific programs that illustrate the program categories.

3.1 Prescriptive CAI

At the computer-as-tutor end of the spectrum are the more traditional computer-controlled CAI programs. Microcomputer-based courseware applications of these kinds are numerous. Materials spanning virtually all school subjects and grade levels are available in the marketplace. Most applications are drill and practice exercises, particularly in mathematics, the physical sciences and language arts.

In mathematics the topics covered include integer arithmetic, fractions, decimals, percentages, modular arithmetic,

prime numbers, factorization, business arithmetic, statistics, graphing, geometry, measurement, equations, algebra story problems, functions, trigonometry and calculus.

In science areas most CAI materials are at senior high school level. Topics covered include: in astronomy -- constellations, solar distances, planetary orbits; in biology -- Mendelian genetics, gene mapping, inheritance patterns, population genetics, circulatory systems; in chemistry -- atomic structure, chemical elements, symbols, formulas, balancing equations, ions, valences, chemical equilibrium, acid-base chemistry, gas relationships, moles, the periodic table, electron structure, bonding, kinetics, electrolytic solutions, oxidation and reduction, stochiometry, quantitative chemistry, organic chemistry; in earth sciences -- climates, climatic patterns, ecological regions, solar radiation; in physics -- linear kinematics, inelastic collisions, the inclined plane, centripetal force, vector analysis, lenses, mirrors, projectile motion, specific heat capacity, calorimetry, vaporization, thermodynamics, standing waves, refraction and reflection of light, electric fields, circuit analysis, and the photoelectric effect.

In language arts the topics include: letters, vowels, consonants, word recognition and meaning, reading comprehension, speed reading, spelling drill, plurals, prefixes, suffixes,

sentence composition, contractions, possessives, vocabulary drill, analogies, affixes, homonyms, homographs, synonyms, antonyms, parts of speech, punctuation, and making an outline.

Other subject areas for which there is a moderate amount of CAI material available include: geography, history, music, and typing. In the area of foreign languages there is courseware covering vocabulary, grammar and usage for French, Spanish, German, Italian, Russian, and Hebrew.

3.2 Educational Simulations and Games

A category of application that goes one step further toward using the computer as a tool is that of educational simulations. These have become very popular as teaching tools in recent years, and programs have been written to simulate everything from the workings of a single lemonade stand to the economic system of an entire nation. Ecological systems consisting of a variety of competing forms of wildlife are also good subjects for such numerical simulations. In a typical application, students are given a certain number of control choices, e.g. how many acres to plant with food crops vs. how many to allot to industrial development, and the computer responds by calculating and displaying the response of the system over the next time period, for example by computing a new value for the per capita income.

Closely allied with such "serious" simulations are computer programs that are written as games, designed to do no more than amuse. We include them here, and shall give examples below, not only because a great deal of programming expertise has gone into their development, but also because, as we have already noted, we believe that there is reason to take seriously the educational potential of such games. For some students these games, even in their present versions, can provide valuable educational benefits, for example, in helping students acquire problem-solving, resource management, and computational skills. A recent doctoral study probes possible educational benefits from the use of computer games by children (Malone, 1980).

3.3 Educational Production Aids

A further step in the direction of using the computer as a tool is taken when one addresses the question of teaching special subjects, such as art and music. For most of the popular microcomputers, programs have been written that teach music notation and theory by allowing the student to enter notes into the computer by specifying their pitch and duration. The notes are then displayed on a CRT screen, and sounded out by a built-in loudspeaker. In this way a line of music can be built up and subsequently replayed in the proper sequence and at normal speed. Most such programs have the ability to handle the recording and

production of several simultaneous voices and provide the student with extensive editing capabilities.

The ability of small computers to produce colored graphic displays with acceptable resolution, combined with such input devices as touch sensitive panels, has been exploited in a variety of ways to permit the user to create artistic effects with a minimum of manual dexterity. Although we know of no programs that have turned this capability to the teaching of art in schools, the potential for taking this step is evident.

The teaching of basic skills, such as reading and writing, points up yet another way in which the computer can be used as a tool. A research project currently underway at Bolt Beranek and Newman, for instance, is experimenting with programs that enable children in the primary grades to "build stories", by choosing which of several paths in a tree-like structure the computer-displayed text should follow. At a later stage in their development, the children are encouraged to write their own "story trees", and to enter them into the computer (Rubin, 1980). Programs of this kind have also been combined with illustrative graphics and with computer-generated speech. The "word processing" facilities of computers can be used to provide related capabilities. In a project at the University of Edinburgh, for example, text editors are being used to enable students to generate both poetry and prose, and to develop a

vocabulary and a set of linguistic tools for producing grammatical and expressive writing (Sharples, 1980).

In another project conducted at Boston University and Bolt Beranek and Newman, a system was developed that employs a transformational grammar to generate a rich variety of sentences of specified form and with specified level of complexity and difficulty (Bates and Wilson, 1980). These sentences are used to provide language practice exercises of several kinds. Artificial intelligence methods are used to create sequences of semantically related sentences. Though this system was designed for use by deaf students, it has application in language teaching for non-handicapped students as well.

3.4 Educational Computer Languages

In all the applications described so far, the computer has been used to do whatever the original programmer meant it to do, and no thought has been given to modifying the program itself. However, it is a common fact of experience that many students very quickly decide that some feature or another of a pre-packaged program is not to their liking, and welcome the chance to modify it. The subject of programming, therefore, first through modifying existing programs, and later in learning to create new programs, often arises in a natural and spontaneous way in the classroom. Acquiring the ability to program has the effect of increasing the versatility of the computer as a tool.

Computers may be programmed, or given instructions, in a variety of different languages, which differ primarily in the types and power of commands that it is possible to express in them. For example, in the so-called "machine language," which is the set of fundamental instructions built into the machine hardware, the set of commands for adding two numbers together includes instructions for retrieving each number from memory, for adding those numbers together in a special "register" (which may have to be set to zero for this purpose) and then for placing the result of the calculation back into memory. It is up to the programmer to remember where (i.e. in which memory locations) he put the two numbers to be added, and where he has put the answer. In contrast, in "higher level" languages the operations can be combined into a single command, chosen to resemble as much as possible the natural language expression "add X to Y, and call the sum Z".

The lingua franca of microcomputers is a language called BASIC, which was developed at Dartmouth College in the 1960's, and was modeled on the earlier standard scientific language, FORTRAN (Kemeny and Kurtz, 1971). More recently, other programming languages, notably PASCAL and LOGO, have been developed and implemented for microcomputers. PASCAL is widely used in professional computing; it is a sophisticated language, not readily acquired by beginning users of computers. LOGO, on

the other hand, was expressly designed for education and, like BASIC, is readily accessible to children for simple tasks. LOGO's procedural control structure, illustrated in Section 1.3 above in the work on turtle geometry, makes feasible the development of extended educational projects involving relatively large and complex programs. The programming concepts and activities underlying this developmental work have very special relevance for the teaching of thinking skills.

The teaching of the set of concepts related to programming can be used to provide a natural foundation for the teaching of mathematics, and indeed for the notions and art of logical and rigorous thinking in general. The major points underlying this approach are as follows.

- It is possible to construct programming languages like LOGO of great expressive power yet so simple to learn that they can be taught effectively to second-grade children.
- Students like to program machines; getting one's program to work is a highly motivating and compelling experience.
- Programming requires rigorous thinking and expression. Students impose the need for precise statement on themselves through attempting to make the computer understand and perform their own algorithms.
- Programming can be used to give students very specific insights into a number of key concepts such as formal procedure, variable, function, and transformation.
- Programming provides highly motivated models for the principal heuristic concepts. It lends itself perfectly to discussion of the relation of formal procedures to

intuitive understanding of problems. It provides a wealth of examples for heuristic precepts such as "formulate a plan", "separate the difficulties", and "find a related problem." Thus, it provides a natural context to make tangible the approach to teaching associated with the ideas of George Polya.

- The concrete form of the program and the interactive aspect of the machine allow "debugging" of errors to be identified as a definite, constructive, and plannable activity. The programming concept of a "bug" as a definite, concrete, existent entity to be hunted, caught, and tamed or killed is a valuable heuristic idea. Debugging of programs provides a valuable metaphor for problem-solving generally, in non-computer contexts as well.

- By enlarging the scope of applications, programming allows every problem to be embedded in a large population of related problems of all degrees of difficulty. For example, through programming, mathematical induction can be presented and generalized by its relation to recursion. Also, the extension of an operation to a larger domain becomes an everyday activity. Generalizing this, generalization itself becomes an activity undertaken routinely by students. Procedures become familiar things that students invent to serve their purposes. They use their procedures as building blocks for constructing more complex functions that often are elements of still larger constructs. Thus through a gradual constructive development, they are able to build relatively powerful models of many kinds.

- The use of computers and programming languages is also relevant to what is perhaps the most difficult aspect of mathematics and science for a teacher: helping the student strive for selfconsciousness and literacy about the process of solving problems. High school students can seldom say anything about how they work towards the solution of a problem. They lack the habit of discussing such things and they lack the language necessary to do so. A programming language provides a vocabulary and a set of experiences for discussing concepts and problems. Programs are more discussable than traditional mathematical activities: one can talk about their structure, one can talk about their development, their relation to one another, and to the original problem.

- Finally, the richness of non-numerical examples open to programming can be exploited to enlarge the cultural base of mathematics and computer courses by bringing them into contact with physical and biological science, language study, geography, economics, and other subjects.

These ideas are further elaborated in Papert (1972). Their use is demonstrated in the context of mathematics curriculum materials (Feurzeig, et al, 1971 and Feurzeig, Lukas and Lukas, 1977). In these materials the LOGO programming language provides the central conceptual framework for both the teaching presentation and the student work.

3.5 Some Observations Regarding Usage

According to a survey conducted for the Department of Education's Task Force on Educational Technology by the research firm of Westat, approximately one out of every four public schools (about 22,000) in the United States now has at least one microcomputer or computer terminal for instructional use by students. The percentage is higher for secondary schools (about 50%) than for elementary schools (about 14%) and intermediate (about 19%) for all other types of schools including vocational, special education and schools containing both elementary and secondary grades. The number of computers and terminals involved is estimated at about 52,000, with microcomputers outnumbering terminals about 3 to 2 (National Center for Educational Statistics, 1981).

Sheingold (1981) examined the use of microcomputers in three different school systems by means of interviews of teachers, students, school and district administrators, technology specialists and community persons, and by direct observations of students using microcomputers. The school systems studied differed with respect to size of student population, number of students making use of microcomputers, purposes for which the microcomputers were being used, extent to which the microcomputers were used in classroom settings as opposed to in resource rooms and other places outside the classroom, roles of teachers with respect to computer usage, and in other ways. All of them, however, had had experience using time-shared computers for instructional purposes as early as the sixties.

In all three school systems studied, the investment in microcomputers and the software to run them was more than incidental. In the largest of the systems (a city of about 800,000), 300 TRS-80s were in use at the time of the study and 300 more were expected by the end of the year. In a one-year period, $600,000 was spend on the design of instructional software for use on these machines. The smallest of the communities involved in the study had a population of about 25,000. In this case the school system had 37 micros (34 Pets and 3 Apples) distributed among seven elementary and two junior high schools.

The uses of the micros varied from system to system and from school to school within any given system. They included drill and practice on basic skills, programming, simulation and games, and computer literacy. In some cases the microcomputers were used in classrooms; however, more typically they were used in resource rooms outside the class.

Sheingold summarized her observations in terms of the five following trends:

- Differential access to microcomputers: In all of the sites the microcomputers appeared to be much more accessible to some students than to others; a fact that was attributed to some degree to the dependence of the use of these machines on local initiative, which differed from school to school. Differential use according to sex was also noted: especially at the level of secondary school, boys tended to make more use of the machines than did girls.

- Emergence of new roles in response to microcomputers: "Teacher buffs" (teachers who promote the use of the computer because of their own interest in using it themselves), and "student experts" who establish instructional roles with respect to other students and collegial roles with respect to teachers.

- Lack of integration into elementary classrooms and curriculum: In only one of the three systems studied were the microcomputers used in classrooms, but even here the ways in which they were used were not well integrated with the curriculum.

- Inadequate quantity and quality of software: Apparently a common complaint among teachers involved in the use of these machines.

- Lack of knowledge of effects and outcomes: Sheingold concluded that no one really knows what the educational and developmental consequences have been for children who have used microcomputers in these educational

settings. Anecdotal reports from teachers tend to focus on social outcomes relating to social interaction, status and self esteem.

Sheingold makes the important observation that some of the schools are beginning to see students who have microcomputers at home. It seems likely that as the accessibility to these microcomputers increases as a consequence of the rapidly proliferating uses of them for entertainment, and especially home entertainment, pressures on school systems to develop curricula and teaching methods that acknowledge their existence will also increase.

A common experience following the acquisition of a computer by a school system has been an unexpected proliferation of uses. Zucker (1981) illustrates this phenomenon in a case study of one primary-secondary school of 800 students. He describes 31 different types of applications that were made of a PDP-11/40 within three years of its installation. These included such instructional applications as the teaching of programming and some aspects of mathematics, the analysis of survey data, automated testing, simulations in physics and social science, and independent student projects. Computers were also used for many other non-instructional (or indirectly instructional) purposes such as games and entertainment, maintenance of mailing lists, production of the student newspaper, record keeping, schedule verification, and so on.

Zucker points out that interest (or lack of interest) in using a computer in an educational context can come from unexpected quarters. Surprisingly, for example, in his case study, mathematics teachers tended not to make as heavy use of computers as did some faculty members in other disciplines. Conversely, a history teacher used to advantage a program that would produce hidden words amidst a matrix of random letters. She used the program as an interesting supplement to her conventional material by hiding words relating to history and writing clues for the hidden words. The puzzles thus produced were dittoed and distributed to the class.

One gets the impression in reading Zucker's account that individual students varied greatly with respect to the extent to which they used the computer and what they gained from that use. Some students were sufficiently interested, he notes, to use the machine on their own time; and some undertook rather ambitious programming projects, one of which was sufficiently impressive to win a first prize in a national student programming contest.

Zucker's report is interesting in illustrating not only the wide variety of ways in which a computer <u>can</u> be used in a school situation but in demonstrating that accessibility and a reasonable amount of support for potential users seems to be sufficient to assure that it will in fact be used in many ways.

4. CURRENT MICROCOMPUTER HARDWARE AND SOFTWARE

4.1 Hardware Capabilities

Several inexpensive microcomputers are currently being produced for the educational market. Five microcomputer manufacturers (Radio Shack, Apple, Commodore, Texas Instruments, and Atari) each produce and sell in excess of ten thousand units per year. Their products are widely available throughout the U. S. (in the case of Radio Shack, at many hundreds of local outlets). Thus one can prepare materials based on any of these microcomputers and have it generally accessible to schools. There are a large number of manufacturers with smaller production and some of their computers have interesting special features, but in taking advantage of such features one is assuming the risk that the computer may be discontinued. This, in fact, was the case with the first significant microcomputer line, the ALTAIR, though the facilities were absorbed by PERTEC Corporation.

Together, the five major microcomputers offer nearly all useful features presently on the market and provide a variety of models and configurations. A typical system in a modest configuration costs from $500 to $1000. This level of system is adequate for many of the educational applications being marketed. Systems with maximal configurations providing very substantial educational capabilities are in the $2000 to $4000 price range.

The five major microcomputers have basic computing elements with roughly the same power, but they vary widely with respect to ways in which the user can interact with them. Usually a configuration offered by a manufacturer can be extended with peripheral devices produced by many manufacturers.

The key components of a complete microcomputer system are: computing element and primary memory, secondary storage, screen output, printed output, input devices, sound output, and possibly, speech input and output.

4.1.1 Computing element (CPU) and primary memory

Computing elements and primary memories are roughly the same across manufacturers. Although inexpensive systems are offered with as few as 4,000 bytes (characters) of memory, for many applications the maximum allowed, usually 32,000 to 64,000 bytes, is required. The effective computing power is roughly equivalent to that employed by a "typical" user on a large-scale time-shared system. This is adequate for most state-of-the-art CAI. In fact, the CPU chips used in current microcomputer systems, together with 64K bytes of memory, provide more computing power than the computers on which much advanced computational research was done until recently. The limitations of present microcomputer configurations for educational work are significant primarily for complex real-time simulations requiring a high

level of visual fidelity (such as is required in flight aircraft control simulation) or with instructional applications employing artificial intelligence methods. Both of these areas, however, represent a quantum jump from the requirements of CAI generally and are even beyond the capability of many present day conventional large computer systems.

4.1.2 Secondary storage

All manufacturers provide a cassette tape recorder interface to give a means of storing programs and data semi-permanently "off-line". Accessing material stored in this way, however, is very slow. Mini-floppy disk systems, which store material magnetically on removable magnetic disks, are so much faster that a program that is too large to fit in primary memory at one time can be kept on disk and brought into memory piecemeal as needed. Mini floppy storage greatly enhances the educational utility of a microcomputer system, and is available for all major types of microcomputers. The rapidly decreasing cost of so-called Winchester disks (see section 6.1.2), which are larger and faster than floppies, is likely to make these the peripheral memories of choice for educational microcomputers in the near future.

4.1.3 Screen output

The standard means of presenting information to the user of a microcomputer system is a television-like CRT (for Cathode Ray

Tube) screen, upon which characters or graphic displays are presented. The amount of information that can be presented varies from 20 rows of 40 characters each to 25 rows of 80 characters. Special characters provide a limited graphics capability in the case of the Radio Shack TRS 80 and the Commodore PET; the other microcomputers have varying general graphics capabilities - they treat the screen as a fine or coarse dot pattern in order to draw pictures. They also have color capabilities that can greatly enhance the presentation. Programs can be written to provide "dynamic graphics" in which varying degrees of animation of pictures can be presented. This feature is difficult to achieve in normal time-shared service on a mainframe computer, inasmuch as it depends on the computer providing data to the terminal on a regular, timed basis.

4.1.4 Printed output

Inexpensive devices that print 40 characters per line on narrow paper are available for a few hundred dollars. If one wants 80 characters per line printed out on normal (8" width) paper or the plotting of pictures as they appear on the screen, the cost of the device is comparable to that of the entire microcomputer configuration ($900-$1500). Color capabilities are still more expensive.

4.1.5 Input devices

The standard input device, provided with each microcomputer, is a typewriter-like keyboard. The capabilities of such a keyboard, if they are skillfully exploited by the program, can permit fairly flexible interactions. Certain keys, for example, may be used by the program to control a "cursor," which designates a position on the screen, giving the user a "pointing" capability. One of the great advantages of microcomputers over large-scale time-sharing systems, is the large number of useful input devices that can be attached to them.

Joysticks, for example, are handles that can be tilted horizontally or vertically, thus providing a simple means of specifying spatial coordinates to the computer. They can be used under suitable program control to identify points on the screen, or to indicate lines, geometric figures, or other objects. Lightpens are touched to the screen to give a position and are used for similar applications but are less reliable. A more expensive device, the "mouse", can be rolled over any convenient surface; signals from a ball bearing on its underside cause the computer to move the cursor on the screen. This allows the user to rapidly sweep to any point on the screen without visually obscuring the display.

For many microcomputers a touch panel is available, which

fits on the CRT screen and enables the computer to determine what part of the screen is being touched by a finger. This permits the user to do such things as specify a choice from a "menu" of alternatives or erase a word or line simply by touching it. The graphics tablet is another X-Y positioning device that translates a pen position on a special "tablet" into coordinates that are sent to the computer. It can be used to point or to draw; advantages of this device include very high-precision and the ability to use it with overlays on the tablet to work directly with finely detailed graphics forms such as maps.

4.1.6 Sound output

Many microcomputers offer, in addition to a single tone that serves in place of a typewriter bell, the capability of producing a variety of tones adequate for the generation of simple music and sound effects, in the case of the Apple, and somewhat more complex music and sound effects in the case of the Atari.

4.1.7 Speech input and output.

Inexpensive ($200-$400) speech input devices are available. Current systems permit a limited vocabulary, typically 16 utterances. To achieve a reasonable level of recognition, they require the user to train the system by recording samples of his mode of speaking each of the vocabulary items. Voice response units are also available that provide a very recognizable, though mechanical and monotone, spoken output.

4.2 Representative Software

The descriptions of programs below are intended to illustrate currently available microcomputer courseware representative of some of the categories discussed above. The programs are classified by general subject area. Most of these programs are available in similar versions on PET, TRS 80, and Apple II systems. Significant exceptions are noted in individual cases. A number of periodicals publish updated lists of CAI programs including brief descriptions and commercial sources (Dresden Associates, 1981; Purser, 1981).

4.2.1 Mathematics

ELEMENTARY MATH by Edu-Disk is a good example of a simple drill and practice program. The subject matter is the four basic arithmetic operations, and the program is designed for the elementary grades. Although the treatment is, by design, repetitive and somewhat tedious, this program shows evidence of careful thought in its preparation, particularly with respect to its responsiveness to student errors.

Like all programs of its type, Elementary Math produces randomly generated problems featuring each of the four basic arithmetical operations, at various levels of difficulty. When the student makes a mistake on an addition problem, for example, the program guides him through the problem slowly, counting out

the two addends to the accompaniment of an ascending scale of notes and the drawing of a row of identical boxes on the screen. Part way through this procedure the program stops counting and asks the student to type the ascending integers until the correct answer is reached. It then repeats the problem. This process continues for ten problems, at which point the problems that were incorrectly answered are repeated once more. Finally, the program reports to the student the percentage that he got correct, and on the basis of that score suggests that he switch to a different level of difficulty -- either higher or lower than the one just completed.

The first section of the program is entitled "Are you ready for arithmetic?". When the student visits this section, the computer draws a certain number of blue and yellow boxes on the screen and asks him whether there are more blue boxes than blue and yellow boxes. If the student answers this question wrong twice in a row, the computer suggests that he is "not ready". The graphics are quite good throughout the program.

HURKLE is an interactive game, designed to introduce students in the primary grades to the definition and use of coordinate systems for designating points in a plane. The game can be played either on a two-dimensional surface or along a straight line. Its value both as a game and as an educational tool is significantly enhanced by unusually good computer-generated sounds and graphic displays.

The object of the game is to find the hidden "hurkle", an imaginary beast located somewhere on the CRT screen (or on the page of a printing terminal). In the more elementary versions of the game the hurkle is confined to a straight line segment, either horizontal or vertical. In this case its position is designated by a single integer, viz. the X or Y coordinate. Alternatively, the hurkle may be free to roam over the two-dimensional plane, in which case its position is designated by X and Y coordinate pairs. The student can specify whether he wants to play the game on a vertical or horizontal line, or on the entire plane, and in the latter case, whether he wants the origin at the center of the screen or in the lower lefthand corner of it. (With the origin in the center the game provides a useful introduction to the concept of negative numbers.)

The method of finding the hurkle is to guess its location by specifying either one or two coordinates, as appropriate. The computer then places a mark on the location so designated, and returns a message giving the approximate direction ("north", "south", "northwest") of the hurkle's hiding place relative to the designated point . No attempt is made to provide coaching or error diagnosis. For example, if the computer says "go west" and the student picks a spot that is east of the previous guess, the error in strategy is not corrected. Performance evaluation and feedback are accomplished solely by counting and displaying the

number of guesses required by the student to solve the problem. The computer also displays the message "You took too many turns" if the number of guesses exceeds a predetermined limit.

The program CHANGE is designed to teach students in the elementary grades how to make change in a store. It produces a set of randomly generated problems by specifying the amount of a purchase and the amount of money offered in payment, and then asks the student how many pennies, nickels, dimes, etc. are required to produce the correct change. The screen displays ten boxes labelled according to the denominations of U.S. currency, from one cent to twenty dollars, and the program requires the input of a one-digit number for each denomination -- even when that number is zero. After the somewhat tedious process of providing these ten numbers is completed, the program asks "do you want to change anything?", and, if the answer is "yes", starts the whole process again. The net effect is extremely boring. If the student makes a mistake, the program so informs him, and asks if he would like to see the correct answer, which it then displays along with his. It then goes on to produce another, similar, problem. No cumulative score is made available, and no graceful way is provided for exiting from the program.

BUGGY simulates the types of errors (bugs) that students frequently make in arithmetic calculations. The user attempts to

find a bug generated by the program. First he looks at the "buggy" answers the program gives to the problems he poses to it. When he thinks he has determined the bug, he tries to demonstrate this by giving corresponding buggy answers to problems posed by the program. This relatively difficult task was designed to give teachers insight into the difficulties that learners often experience in acquiring a correct procedure and the enormous variety of plausible faulty procedures that they may develop. At present BUGGY is available only on the Apple II.

DARTS was designed to teach fractions to elementary math students (Dugdale and Kibbey, 1975). The screen displays balloons at random places on a number line that has specified end points (e.g., the interval from 2 to 3). Students guess the positions of the balloons by typing in fractions or mixed numbers. Following each guess an arrow travels across the screen to the position specified by the student. If the guess is correct, the arrow pops a balloon; otherwise the balloon remains on the screen with corresponding numerical value displayed. The student's task is to pop all the balloons.

In a deductive logic game called BAGELS, the student tries to guess a number with a specified number of digits (typically 3, 4, or 5). After each guess he is told, in effect, how many digits of his guess are correct in both value and position, and how many are correct in value only. The objective is to teach

correct inference, use of partial information, and guessing strategies, with the aim, for example, of minimizing the number of guesses.

NUMBERGUESS is a simpler number guessing game. The student tries to guess a number generated by the computer, usually between 0 and 100. After each incorrect guess, the program responds either "too high" or "too low." The object is to learn to use this feedback increasingly efficiently to reduce the bounds of the search and improve the guess (and possibly, to teach the optimal strategy for minimizing the number of guesses -- binary search, i.e. halving the interval of uncertainty on each trial).

4.2.2 Language Skills

HANGMAN is a well known pencil-and-paper game that has been implemented on many different microcomputers for use in teaching spelling and word recognition. The computer randomly picks a word from a previously stored list (which may be generated either by the teacher or by another student), and the student tries to guess it by picking letters one at a time. The student is told after each pick whether that letter is contained in the word. The computer starts the game by displaying a picture of a gallows and a set of dashes, each of which stands for one letter of the desired word. When a letter picked by the student is contained

in the target word it is displayed by the computer in its correct position. For each selected letter that is not contained in the word, a part of a human body is drawn on the gallows, starting with the head. The object of the game is to guess the word before the picture of the body is complete.

MIXUP is a similar game in which the student is given a set of letters that, when put in proper order, will spell a word. The object of the game is to rearrange the letters to form the word. In another game, called CROSSWORD, the computer uses a stored list of words and their definitions to produce a crossword puzzle, using as many of the words as possible. In both these programs, as in HANGMAN, the list of words used may be specified either by the teacher or by another student.

4.2.3 Social Sciences

CIVIL is a simulation program based on famous battles of the American Civil War. The student chooses one battle from a list displayed by the computer at the start of the simulation. He is given a brief description of the historical circumstances surrounding the battle he has chosen, and then is asked to allocate his limited resources, in the form of money, to satisfy various conflicting needs. These include salaries, food, and ammunition. The student may compete against the computer or against another student. In the latter case, the two students

choose sides and are allocated initial sums of money and men in proportion to the historical data. The program contains a very simple mathematical model designed to predict the outcome of the battle, based on the resource allocations made by the players. The simulation model used is not revealed to the students, however, and does not allow them to modify their strategies during the course of the battle. The entire outcome is determined by the initial choices of strategy, and no guidance is given for improving that strategy.

In LEMONADE the student is asked to make choices regarding the management of a simulated lemonade stand. He must decide how many glasses of lemonade to make on any given day (he may be given a weather report), how many signs to erect, each of which requires a certain capital outlay, how to advertise his product, and what price to charge. After receiving the student's inputs, the program computes how many lemonades have been sold, based on a model that is not directly accessible to the student. It also computes the incremental and cumulative profit or loss of the enterprise. No hints or other form of coaching are made available to the student, and no standards are given by which he can judge his performance.

In an economics management game called MARKET, two students take the roles of rival company executives competing for the market for the same product. The players are assigned the fixed

and variable production costs for marketing the product, the number of items that will be sold with no advertising at a given price, and the initial values of product inventory, cash on hand, and total assets. Each player makes quarterly market decisions on the values of production level, advertising budget, and unit price of his product. The program then reports for each company the quarterly profit, percentage share of the market, cash on hand, number of units sold, inventory, and total assets. The companies are penalized for overproduction by a 5 percent inventory charge for stock on hand. If there is negative cash on hand, there is a 5 percent charge on that amount of debt. Events occur along the way that dictate changes in marketing strategy. The game ends ends when a company either goes bankrupt or accumulates a specified level of total assets.

HAMMURABI was designed to teach some of the basic principles of economic management as applied to a simple agrarian economy (Wing, 1965). The student takes the role of ruler of ancient Sumer. He receives semiannual reports of harvest yield, population, area of farmland under cultivation, inventory, and standard of living. He must decide how to use the country's agricultural resources -- how much grain to allocate for food, how much to plant for the next crop, how much fertilizer to use, when to change the acreage and/or density, irrigation policies, and so on. The situation is dynamic -- there are major increases

in population to deal with, problems of expansion entailing the acquisition of new land, technological innovations such as plowing and letting soil lay fallow, and disasters resulting in grain destruction, which alter previous decisions and policies. Although HAMMURABI's responses to the student's inputs are based on a reasonably realistic model, the program does not provide any explanation or rationale for the outcomes along the way. The student is not helped to understand why things turned out the way they did.

CRIMEX is a somewhat more complex simulation dealing with the problem of urban crime. The student is given statistics on the annual number of violent and non-violent crimes in each of five precincts of an imaginary city, as well as information regarding the per capita income and seven other "social indicators" for each precinct. He is then offered an opportunity to change the conditions in each precinct, and the computer, using a simulation model not directly accessible to the student, "predicts" changes in the crime statistics. The program has built-in checks that prohibit "drastic" changes (a term left undefined) in any of the social conditions. A major criticism of this simulation, and others like it, is that because it treats only a very small portion of the problem of running a city, it tends to ignore or gloss over the costs associated with the resource allocations adopted by the student, and runs a serious risk of trivializing the problem it is intended to illuminate.

Another urban social science simulation, entitled RATS, incorporates a model for the population dynamics of an urban rat population as it is affected by a variety of control measures. Students are able to specify the type and quantity of poison to be administered, as well as the level of garbage to be permitted on the city streets, the latter quantity measured in arbitrary, undefined units. The program computes such quantities as total births and deaths in the rat population, immigration and emigration rates, and direct and indirect costs of the rat control program.

In STABIL, the student attempts to stabilize the U.S. economy, using such tools as changes in the level of federal spending, tax policy, changes in the money supply, and wage and price controls. The program computes quarterly changes in GNP growth, unemployment rate, inflation, profits, and interest. This program is marred by a poorly designed user interface, and perhaps more fundamentally by the enormous difficulty of the task of simulating a national economy. Like most programs that simulate very complex systems for well intentioned educational purposes, this one runs the risk of over simplifying the subject matter and conveying an erroneous impression of the reliability of economic projections. In addition, by not taking into account the effects of economic policy on income distribution and a host of other variables, it fails to capture the complexity of economic decision making.

LIMITS uses a simple world model comprising five global subsystems (population, pollution, food supply, industrial output, and resource usage). These are linked together by six rate variables (birth rate, death rate, pollution rate, resource usage rate, industrial output growth rate, and food production rate). The student investigates the five subsystems by making changes in the initial values of the rate variables and specifying the times these changes are to occur. The program then graphs and/or tabulates the values of world population, food supply, average income, and death rate from the initial year to a specified end year in 25-year increments. This program is typical of a number of poorly conceived "educational simulations." The notion that complex global phenomena can be treated by such simple models is naive and misleading. Even accepting them as educationally useful abstractions, these programs provide no insight into the operation of the model. The interactions among the variables are not obvious and no attempt is made to show interaction effects.

4.2.4 Natural Sciences

In a program named ODELL LAKE the student learns something about marine ecology by taking the role of a fish and attempting to survive and prosper. At the start of the program, the student is asked to choose which of six species of fish he would like to be. He is then informed that he has encountered one of the other

species, and is given a choice of optional actions, which include chasing the newcomer away, trying to escape, trying to eat it, or simply ignoring it. Through trial and error the student learns which fish he can safely ignore, which pose a threat, which are good to eat, and so on. Encounters with species other than fish are also possible, and the student learns which fish are particularly vulnerable to predation by land animals or birds, which eat insects or plankton, and which are essentially isolated from the environment outside the lake. This program is characterized by extremely well realized animated graphic displays, which contribute very significantly to its educational appeal. It is much more exciting, for instance, to watch two animated fish fight on the CRT screen than to read the message "you have been eaten".

The program QUAKES is designed to teach the student the fundamentals of seismology and earthquake prediction. The material is poorly organized, however, and inadequate use is made of graphic displays. The general impression given is that the subject matter is too complex for the student to understand. The program does not provide enough feedback for the student to signal when he does not understand something, and as a result it is likely to confuse and discourage him.

MILLIKAN is a simulation of the famous Millikan Oil Drop experiment for measuring the charge on the electron. It is also

a good example of how not to use simulations in the classroom. No graphics are used, so whatever picture the student has of the actual apparatus must come exclusively from written material that is used in parallel with the computer. The computer simply displays a mass of numbers intended to denote the velocities of various drops as measured with the apparatus. The student inputs different values of the potential between the parallel plates in the apparatus, to produce differing values of the retarding electric field. When he does this the velocities displayed change in nonobvious ways. Sooner or later, if he is patient, the student may succeed in finding a value for the inter-plate potential that causes one of the velocities to vanish. At this point he can ask the computer to calculate the value of the electron charge, and he gets a numerical value, in Coulombs, which is displayed to ridiculously high numerical precision, giving no indication of experimental uncertainty. The entire experience is mystifying and vague, even to a highly expert user, and is a good example of the misuse of computer-based simulation for teaching.

THREE MILE ISLAND simulates the operation of a pressurized water nuclear reactor. Evidently intended primarily for recreational purposes, the program can nevertheless be used effectively in the science classroom. It runs only on the Apple II computer. The student's task is to monitor and control the

reactor while maximizing its electrical energy output. Displays are available to help in monitoring the containment building, turbines, filters, condenser and reactor core. The student can open and close valves, control pumps and turbines, clean filters, and position control rods. He must carry out regular preventive maintenance on critical components. Demand for electric power may vary, necessitating adjustments. Problems may develop such as failures of components or instruments. The student must respond with the correct procedures. In some situations there is the possibility of a melt down. Although the numerical model governing the behavior of the reactor is far from realistic, this program, which has become a "best seller", features good use of the Apple's high resolution graphics capability, and an effective use of conflicting goals (safety vs. availability of the power plant) in order to generate interest.

BUFFALO is a simulation in which the student investigates the effects of alternate policies of harvesting on the growth of a herd of buffalo. The student starts with a population comprising specified numbers of male and female adults, yearlings, and calves. A standard task is to increase the herd size to a fixed level and then halt further growth. Two policies of herd management are available: total freedom of herd control, and automatic safeguards to protect the herd against excessive harvesting. The student may intervene each year of the study to

harvest selected numbers of specified elements of the herd, e.g., female calves. The results (the size of each component of the herd) are computed and plotted along the way.

GENE1 is a simulation of the inheritance of genetic traits built upon the Mendelian assumptions that genetic traits are independent and non-sex-linked, that each allele of a trait is equally likely to be transmitted, and that one allele is dominant and the other recessive. The student chooses the two traits to be studied, states which is dominant, gives the genotypes of the female and male parents, and specifies the number of offspring to be observed. The program computes the genotype and phenotype of each offspring (if there are not too many) and the genotype and phenotype ratios. The program can be used to simulate Mendel's original experiments and to explore possible genetic inheritances within successive generations of a family.

4.2.5 Educational Games

The most spectacular use of graphics so far available on microcomputers is to be found, not in programs designed explicitly for educational purposes, but in games. This is understandable, inasmuch as the market for computer-based games is large and growing, and the enjoyment of these games can be greatly increased by the appropriate use of high resolution color graphics. In commercial jargon, there is a high "value added" to

computer graphics in the recreational game market. However, precisely the same point can be made with respect to programs whose primary purpose is to teach, rather than to amuse. Educational software is also significantly improved by graphic displays, as we have already noted in the case of several programs described above. We therefore feel that it is useful to include in this paper descriptions of recreational games featuring particularly good uses of graphics, if for no other reason than to point out the glaring discrepancy between them and educational programs.

The game of STAR RAIDERS has probably the best graphics yet implemented on a microcomputer. Written for the Atari 800 machine, it features a fully animated, high-resolution color display that simulates the view from the control room of some future rocket ship travelling through space. Objects appear to originate at the center of the screen and stream outwards toward the edges, giving an irresistible impression of forward motion. This "motion" may be controlled by joy sticks -- hand-held levers that may be manipulated forward and back and from side to side, to alter the orientation, and thus the velocity vector, of the rocket. Laser guns are available to the player, and are controlled by buttons on the joy sticks. Their operation gives rise to flashes of colored light seemingly emanating from the "wings" of the imaginary spacecraft, accompanied by whooshing noises straight out of a science fiction movie.

From time to time, deadly enemy rocket ships equipped with advanced energy beam weaponry appear on the screen and must be dealt with by the player in a matter of seconds or they will penetrate his energy shields and destroy him. Random meteorites and other space debris must also be evaded or destroyed. All this takes energy as well as skill, and the former commodity may be replenished at star bases, of which there are several sprinkled throughout the galaxy.

The player may at any time request an overall view of the galaxy, in the form of a continuously updated star map showing the locations of all star bases, as well as enemy ship concentrations. But the star bases are also under attack by the computer controlled enemy, who grows more numerous and resourceful as the player's skill level increases. Thus, a complicated -- and at times agonizing -- resource allocation problem must be solved in "real time": whether to allow a base to be destroyed, or undertake to rescue it against overwhelming odds.

Eventually, if the player is successful, the galaxy is rid of the invaders, the Federation is saved, and the player, who starts as a "Garbage Scow Captain", is promoted in rank. He may have difficulty attaining the highest rank, however, as the demands made on his dexterity, eye-hand coordination, and decision-making skills become increasingly difficult to meet.

Interest and enjoyment are maintained throughout the game, however, largely by the remarkably "realistic" sights and sounds provided, as well as by the carefully graded levels of difficulty that ensure that the challenge is never so great as to become frustrating to the player.

Another compelling game involving interaction in the fantasy world of deep space is STARTREK, a resource management decision-making game. This game originated on large computers, but versions of it exist for virtually every microcomputer on the market. The student's task is to command a spaceship (the starship "Enterprise") to seek out and destroy all the enemy ("Klingon") ships in the universe. Conditions and constraints of several kinds apply -- on the total mission time; on the number and effectiveness of the starship's several types of weapons and protective devices; on the starship's capabilities for gathering intelligence; on the location of enemy ships, refueling bases, and asteroids; on the energy and time costs of alternative navigational actions, both global and local, and so on. The student thus has a large set of decision options involving information gathering, navigation, offensive and defensive tactics, weapons utilization, firing priorities, among others. There are relatively straightforward mathematics problems to be solved, such as determining effective firing angles and evasive actions, given coordinates of the targets. Also, the student has

the problem of discovering the nature of the enemy's navigational and weapons capabilities -- these are not specified and must be inferred. Finally, there are the difficult problems of allocation of time, fuel, and weapons resources involving issues of optimization and mixed strategies.

We have already mentioned, in the Introduction to this report, the computer-based fantasy simulation games, which grew out of the familiar role playing game Dungeons and Dragons. These games involve both language and logic skills, and depend on written descriptions of objects and situations, rather than detailed graphics, to interest and involve the player. The original fantasy simulation game, ADVENTURE, involves the player in a series of struggles against a variety of monsters, wild beasts, and other dangers, his only weapons being those he finds along the way. Many of these weapons possess magical powers the exact nature of which is not revealed and must be deduced by trial and error. The ultimate goal is the accumulation of treasures which abound in this fantasy world, but are heavily guarded by its inhabitants. It is in the honing of the problem solving skills necessary to defeat these guardians that the real educational value of the fantasy simulation games resides.

5. EVALUATIONS OF EDUCATIONAL EFFECTIVENESS

There have been very few well-designed and carefully-controlled formal evaluations of the use of computers for instructional purposes. Indeed there is a paucity of exemplary models of educational evaluations and a lack of theoretical work to provide a sound foundation for such studies. Thus, it has recently been stated that there is no scientifically conclusive evidence that CAI is instructionally effective (Moshowitz, 1981). The same could be said, however, about most educational evaluations involving non-computer-based methods and treatments.

A recent review of CAI developments and trends cite a few prominent evaluation studies that show remarkably consistent results (Chambers & Sprecher, 1980). The studies reviewed include:

- A tutorial CAI course in physics developed at Florida State University (Levien, 1972)
- A CAI course in Russian developed at Stanford University (Levien, 1972)
- Tutorial CAI courses in mathematics and reading involving 12,000 fourth through eighth grade students in the Chicago schools developed by the Chicago City Schools Project now being formally evaluated by Educational Testing Service (Sugarman, 1978)
- CAI materials developed for community college courses in accounting, biology, chemistry, English and mathematics, and administered using the University of Illinois PLATO CAI system (Murphy & Appel, 1977)
- CAI materials developed for community college courses in

mathematics and English using the TICCIT CAI system (Alderman, 1978).

The general conclusions from these and other recent evaluations involving elementary school through college programs were that the use of CAI either improved learning or showed no differences when compared to a traditional classroom approach, and the use of CAI usually reduced learning time. However, when there was limited faculty involvement with students, as in the TICCIT-based mathematics course, for example, completion rates for the CAI course were considerably lower than those for the traditional course.

Another review of seventeen evaluation studies generally found that traditional instruction is more effective when supplemented by CAI (Edwards, Norton, Taylor, VanDusseldorp, & Weiss, 1974). The same review, however, found that in eight out of the seventeen CAI studies, there was no significant improvement in performance by CAI students over those receiving traditional instruction. However, CAI did generally reduce learning times. Also, CAI was relatively more effective for low ability students than for those with average or high ability. A recent doctoral study probes possible educational benefits from the use of computer games by children (Malone, 1980).

There is very considerable interest in methods of effective teaching of programming concepts and skills. However, a review

of studies involving instruction in programming found that most studies had an inadequate experimental design (Ragsdale, 1978). Flaws commonly found included non-random assignment of subjects, the assumption that different groups received identical instruction, experimenter bias, use of "available resources," short experimental treatments, home grown tests, and questionable analysis techniques. Thus far there are no definitive research investigations in this area.

Research investigations of the use of programming concepts and work in the teaching of other subjects are still rare. Most studies have involved small numbers of students over relatively small time intervals. In one of the earliest studies, LOGO programming was used for teaching algebra to mathematically average seventh grade students, two years earlier than for their ability peers (Feurzeig et al, 1969). Performance scores on a standardized test were not significantly higher than those of a control group, but follow-up studies showed improved performance in subsequent mathematical work of most subjects relative to their peers. A later study involving LOGO programming found improvement in children's scores on an unstandardized mathematics test (Milner, 1973). Another study involving the use of BASIC programming in mathematics teaching found that numerical ability, particularly among "less able" children was improved (Bjork, 1975).

More recent work at the University of Edinburgh investigated the teaching of mathematics through LOGO programming and again found improvement in the mathematical understanding of less able children as well as general improvement in mathematical communication and articulation of all experimental students (Howe, O'Shea, & Plane, 1979). In an MIT-based project, sixth grade children used LOGO to write programs for drawing geometric figures and graphic designs. Their LOGO work exhibited great diversity in cognitive approach and learning style and provided a rich framework for analysis of their conceptual developments (Papert, Watt, DiSessa, & Weir, 1979). An informal case study exploring the effects on teacher trainees of using LOGO to investigate topics in elementary mathematics found marked improvement in their understanding of the material as well as enhancement of their enjoyment and of their ability to make personal mathematical discoveries (du Boulay, 1978). Another recent set of case studies exposed a number of teachers to a variety of computer-based instructional approaches. The teachers were most favorably impressed with the pedagogical potential of computers when they are used by students as developmental tools, in activities such as LOGO programming projects (Olds, Schwartz and Willie, 1980).

6. NEAR-FUTURE TECHNOLOGICAL CAPABILITIES

Although the primary purpose of this paper is to review current trends in computer aided instruction, computer technology is changing so rapidly that it seems imperative to try to make some prediction of where it is going and how this is likely to affect education over the next few years. In approaching this subject, it is useful to make a distinction between "hardware" -- computers, disk drives, terminals and so forth -- and "software", which is the generic term for the programs that contain the instructions to the computer, and cause it to do what the user wants. From the standpoint of the teacher interested in the potential of the computer in education, it is clearly the software that is of greatest importance. However, questions of hardware capability can have an impact on the feasibility of various kinds of educational software. Moreover, a major trend in computer technology has been the steady reduction in its cost, a reduction made possible almost entirely by improvements in fabrication and design techniques. This trend, which is expected to continue, is one of the underlying motivations for the present review.

6.1 Hardware Trends

In sharp contrast to most other products in an inflationary economy, computer hardware has been steadily decreasing in price.

This can be viewed in two complementary ways. On the one hand, the same amount of "computing power" (and we shall describe in greater detail below what this term means) can be bought for less money; conversely, the same money buys more. We shall start this discussion by describing what improvements are currently being made in computer capabilities, and what their implications may be for education. Prices are harder to predict with any certainty, but some plausible forecasts may be made by an extrapolation of recent trends.

For our purposes it is convenient to divide a computer system into the four basic units described above in Section 4.1: the Central Processing Unit (CPU), a microprocessor that receives, interprets, and executes the instructions; the memory, which may be in the form of an integrated circuit "chip", known as "on-board", or "core" memory, or in the form of so-called "peripheral" memory; the input devices, such as keyboards and tape drives; and the output devices, such as printers, plotters and graphics displays. We shall deal with each of these component types in turn.

6.1.1 The Central Processing Unit

The microprocessor, also known as the central processing unit, or CPU, is the heart of any computing system. In most microcomputers, it is located on a single "chip" of silicon about

an inch square (a feat undreamed of as little as 15 years ago). Almost all the microprocessors used in microcomputers today are capable of handling eight bits (ones or zeros) at a time. This number of bits (called a "byte") is enough to encode a single alphabetic or numeric character. It is not enough to contain an entire machine instruction, however. This means that microcomputers must spend time shifting bytes around, in order to perform relatively simple tasks. One of the trends that is almost certain to overtake the microcomputer industry is the adoption of microprocessors capable of handling two bytes (16 bits) at a time. Such 16 bit processors exist today, but they are relatively new and therefore also relatively expensive. Moreover, in most cases they cannot run the very extensive software that has already been written for 8 bit machines. This situation is only temporary, however, and it is likely that 16 bit machines (and possibly even 32 bit machines) will begin to dominate the microcomputer world sometime within the next five years. The enhanced computing power that this will represent will mean, among other things, that sophisticated, animated color graphics will become feasible at very low cost. It will also allow the development of sophisticated programs capable of interacting with the student at a very high level.

6.1.2 The Memory

The trend in computer memories is simply stated: more for less. A few years ago the largest memory chip available held approximately 16,000 (abbreviated 16K) bytes. Today 64K chips are becoming available, both in the US and in Japan. As fabrication techniques improve, the price of these units may drop to no more than a few dollars each, making them appropriate for use in even the least expensive microcomputers. Since 64K bytes are all that can be used efficiently by the current 8 bit processors, the day is not far off when this will become the standard memory size for all microcomputers.

Peripheral memory devices are also undergoing rapid changes. The floppy disk is beginning to drive out the cassette recorder for most educational uses, mainly because of its far greater speed. Recent techniques such as increasing the density of information storage, and the use of both sides of the disks, have quadrupled the storage capacity of these disks, which can now store dozens of average-sized programs. A floppy disk drive currently costs about $500 dollars, but this price is likely to be reduced by half over the next few years. Larger computer systems today typically use a different kind of disk, called a Winchester, which can store more information (millions of bytes) and retrieve it approximately ten times faster than floppy disk drives. It remains to be seen whether these disks will become

popular for microcomputers. Their popularity is hindered at present by their relatively high cost, as well as the noise and heat they produce.

Other technologies that are likely to have an impact on the microcomputer include videodisks and bubble memories. Each of these currently has a capacity comparable to that of the Winchester disk, though each has the potential for much greater storage. Videodisk technology, which can store an entire hour-long movie on a single disk, has the disadvantage that it is a "read only" medium -- information cannot be stored on it by the computer. It has the advantage, however, that the disks are cheap and almost indestructible. Bubble memories are potentially very rapid access read/write devices, but are still too expensive to be practical in many applications.

6.1.3 Input Devices

The increasing ability of computers to recognize spoken words has given rise to yet another form of input device that is particularly attractive when dealing with young children who cannot type. Speech recognition modules are presently available "off the shelf", in the price range $400 to approximately $2500 that can reliably recognize from 16 to 100 different spoken commands, after a brief "training session" with the speaker. The best of such devices are still too expensive to be realistic for

use in schools, and moreover are somewhat bothered by background noise, but the price should drop once the development costs have been recovered, and the use of special microphones and improved recognition algorithms will probably solve the background noise problem.

6.1.4 Output Devices

Output devices are also coming down dramatically in price. Today the printer is often the single most expensive component in a microcomputer system. While this is likely to remain true over the next few years, at least for high quality printers, a variety of innovative printers have recently come on the market. Although they do not produce "letter quality output," they are quieter and faster than impact printers, such as typewriters, and cost under $1000. It is to be expected that such printers will eventually cost no more than a portable typewriter--under $200 in today's dollars.

Although devices exist for producing hard copy of graphic material, they are still quite rare on microcomputer systems, largely because of their cost. For the near future the production of such graphic output may be achieved most efficiently by simply taking an instant picture of the terminal screen. However, the demand for plotters in industry and research may be strong enough to produce economies of scale and

reduce the price to the point at which they would become practical for educational use.

A very attractive option, especially for educational applications, is voice output. As Texas Instruments' Speak and Spell and related devices already demonstrate, relatively small machines can now be programmed to speak about 100 words quite intelligibly. Texas Instruments sells a voice production module, modeled on its Speak and Spell, that attaches to its home computer and can be programed by the user. Other vendors are beginning to introduce speech output systems with much larger vocabularies and greater intelligibility and quality. The attractiveness of "talking machines" in education has been amply demonstrated by a number of devices now on the market, and it seems only a matter of time before the computer will take full advantage of the possibilities in this area. For example, the first text-to-speech systems that can handle arbitrarily large vocabularies have become commercially available this year. These systems make mistakes in handling some text inputs, and their speech output quality is not as good as that of some of the systems with fixed vocabularies. However, their performance can be expected to improve steadily.

6.2 Software Developments

"Software" is the generic term for the instructions that cause the computer to do what the programmer intends. At the most fundamental level, these instructions may be thought of as collections of ones and zeros that are directly interpretable by the computer's microprocessor, and cause execution of quite simple commands, e.g. store the number (also written as a string of ones and zeros) into a certain location in memory, read a number off the floppy disk drive, print an alphabetic or numeric character stored in the processor on the terminal screen. However, writing a program of any complexity in this so-called "machine language" is extremely cumbersome, and takes quite a long time -- much of it spent finding errors, a process known as "debugging". From the earliest days of computers, therefore, a great deal of attention has been given to making it easier to program them, with the result that today very few users of computers even know the machine language that actually runs the programs they write. Instead, they deal with the computer through a bewildering variety of software aids, collectively known as "utility programs", that translate high level commands, written in languages designed to approximate ordinary English, into the long sequences of elementary instructions that the computer is capable of executing.

6.2.1 High Level Languages

The utility programs most obvious to the programmer are those that enable him to use a particular higher level language, such as FORTRAN, BASIC, or PASCAL. Some of the improved software capabilities to be expected in the next few years have to do with the availability of increasingly powerful languages on a wider variety of microcomputers.

Differences among high level languages are not trivial. The ease with which programs can be written, de-bugged, modified, and combined with other programs, can vary considerably from one language to another. In addition, the absence of an agreed upon standard form for certain languages (notably PASCAL, and even BASIC), can adversely affect the portability of programs, i.e. their ability to run on more than one type of computer. Finally, to the extent that one wishes a student to learn programming as part of the educational enhancement that the computer makes possible, the choice of programming language can be critical.

One of the mini-revolutions that has occurred in recent years in the field of software development has been the rise of a programming technique known as "structured programming". It would transcend the purpose of this paper to go into detail as to what is implied by this term, but an important aspect of it consists of analyzing the overall goal of a given program into

manageable sub-goals representable as procedures, and linking these to form larger "super-procedures", until the final program performs its intended tasks. Writing programs in this fashion can result in substantial increases in productivity, in addition to creating more usable and more flexible final products. With the increasing availability of languages such as PASCAL, LOGO, C, and the newly created ADA, which are explicitly designed to encourage structured programming techniques, the hope is that much of the drudgery and inefficiency of program creation can be reduced or eliminated. This should make possible the creation of advanced educational tools at a fraction of today's cost. We hasten to point out, however, that this development will only make available a powerful tool -- it obviously cannot by itself assure that the tool will be properly used.

6.2.2 Operating Systems

High level languages are not the only software tools used to make computer programming easier. Even after a program has been compiled and is in a machine-readable form, it is incapable of performing its functions without the help of another software package called the operating system. This is perhaps the most basic program in any computer system. It contains special purpose program packages, or subroutines, that operate the various input and output devices attached to the system; provides access for the program to special purpose hardware, such as real-

time clocks; and also contains the user interface, including such functions as access to user-created files, database management, and even the echoing of characters typed at a terminal.

Just as there are many different languages, so there are many operating systems. Among the most popular for microcomputers, at present, are the CP/M and OASIS systems. However, this too may change, and more powerful operating systems, such as Bell Laboratories UNIX system, originally designed for mainframe machines, are now beginning to become available on microcomputers as well. Inasmuch as operating systems are specifically designed to be "invisible" to the programmer, the major advantages that this trend will produce will be more extensive facilities for data management and for communication between different processes running on the same machine. It is also quite likely that such operating systems will come equipped with improved software for handling inter-computer communication, thus allowing access, for example, to extensive data bases on large, remotely located computer systems.

6.2.3 User Interface

The problem of making computers "friendly" enough to be run by students and teachers who are unfamiliar with the details of how to program them, and may wish to remain so, is a difficult one that is receiving an increasing share of attention as falling

computing costs make possible ever more ambitious "user interfaces". One approach is to have the computer approximate a human being in its ability to understand natural language commands. Current computer science research has made important strides in this direction, but capabilities are still very limited. The situation at present seems to call for giving the computer other kinds of capabilities that will allow it to be responsive to nonlinguistic, but natural, human input. Thus the use of the touch panel or the mouse, as described above in Section 4.1.5, enables programmers to display a "menu" of possible commands, and to have the user designate the one he wants executed.

One development that appears likely in the next few years is the design of generalized user interface tools that can be adapted for a wide variety of different programs running on different machines. These, in effect, would be specialized high level languages, specifically suited for providing a natural mode of communication with the computer. It is likely, in fact, that one or at most a few of the more popular of these tools may become a de facto standard, thus providing a standardized method for communicating with computers in very different contexts.

6.3 Artificial Intelligence

Within the specific area of CAI there will surely be significant advances in artificial intelligence methods for instruction and continued extension of intelligent CAI capabilities on educational computer systems. Intelligent CAI (ICAI) systems include subsystems for simulation, language processing, explanation, diagnosis and guidance. These are often large and complex programs requiring extensive computation. Typical components of an advanced ICAI system include the following.

- Natural language front end. This capability allows the student to use relatively unconstrained English input when posing questions. It facilitates ready access to the system and improves the information flow between the student and the program. This is particularly valuable in applications involving relatively large and open task domains in which the discourse forms are not tightly stylized.

- Domain-specific knowledge representation. ICAI systems require extensive knowledge about the problem domain. For example, in qualitative chemical analysis problems, there must be a knowledge base including data about chemicals (substances), tests (measurements), known properties, potential unknowns derivable from these known properties, and plausible measurement strategies. Clearly, the organization of such a complex hierarchically structured, dynamically changing knowledge base is crucial to the effective operation and extension of the system. Information in the knowledge base should be easy to access, extend, and modify, both by the AI expert program and by the human experts designing or using the system.

- Multiple levels of simulation. The system whose operation is being studied is typically composed of subsystems that may also be composite. When this

complex system is simulated within an ICAI program, it is necessary to communicate important information about the system's behavior at different levels of abstraction.

- Explanation facility. The system must not only be capable of monitoring and modeling its own behavior, but also in explaining this behavior to the student in an articulate manner.

- Dual models of the instructor and student. These are required to diagnose the student's difficulties in a very specific deep fashion and to provide insightful guidance.

These capabilities, in their most developed forms, are products of technologically advanced research and programming work. The total number of such systems is small at present, and largely limited to advanced professional and technical training. The experience with the educational use of ICAI systems has been limited, and their potential educational benefits are largely unevaluated. Nevertheless, current ICAI systems demonstrate many technologically compelling capabilities and much of the intellectual capital of CAI research is dedicated to further development of such systems (Gable and Page, 1980).

The methods used to exercise these capabilities have generally required substantial computational systems of the kind available in major research laboratories, not in schools. However, because of the continuing rapid rate of advance in hardware technology, today's research systems will be the inexpensive personal computers of the next five to ten years.

Some current microcomputers already provide enough processing power and memory capacity to accommodate certain ICAI systems, albeit with some compromise in their scope, depth and speed of operation. A restricted version of the BUGGY system has been implemented on the APPLE II microcomputer. Even more impressive are the plans for implementing the STEAMER ICAI system on a large Z-80 based microcomputer, at least in a restricted version.

STEAMER is a knowledge-based system designed to teach the operation of steam propulsion plants. STEAMER has strong capabilities for generating coherent articulate explanations of the power plant and of its components and subsystems. It animates diagrams of the systems and elements to show vividly the operation of the power plant in response to student inputs and in concert with its own explanations. It employs incremental qualitative simulation methods to model the changes in each component of the system in response to external changes made by the student. STEAMER is currently implemented on a Digital Equipment Corporation 11/70, a large minicomputer system. The program is large; the simulation and animated display require high speed numeric and symbolic computation. Implementation of this system on a microcomputer, which is currently underway will demonstrate that instructional systems with significant ICAI capabilities will be available on microcomputers in the near future.

The ICAI advances are the most dramatic developments anticipated from a technological point of view. It may well be the case, however, that another set of developments, relating to the extended use of programming languages in education, will be even more important intellectually.

6.4 Programming Languages

Early use of programming languages in education was directed to teaching computer programming as a special technical or vocational subject. Later it became apparent that programming languages were superb vehicles for teaching mathematics, science, language arts, and other standard school subjects. The related notion that programming languages provide unique capabilities for teaching problem solving and thinking skills is still more recent and only beginning to be deeply explored. Some arguments for this thesis are presented in section 3.4, above.

The initial work in using computers and programming to teach mathematics and computational methods was done with FORTRAN, a programming language designed for computational rather than educational applications. A FORTRAN-derived language, BASIC, was the major language developed for use in teaching. BASIC is an easier language than FORTRAN and is well suited for development of relatively simple programs. BASIC has thus far remained the dominant educational language.

More recently, other programming languages that have been introduced for use in education include APL, SMALLTALK and LOGO. Each claims special capabilities and benefits. APL is especially appropriate for teaching computational algorithms in numerical mathematics, particularly at the undergraduate level. SMALLTALK provides sophisticated capabilities for animated graphics and simulation programming, but is not an easy language to learn. LOGO, because of its simple procedural syntax, is a particularly appropriate language for work with young children. The use of the LOGO-controlled robot turtle and turtle graphics, enhances its effectiveness in elementary classroom work. The extensional features provided in LOGO and SMALLTALK give students the expressive power required for developing large, complex programs in extended project work, at all levels of sophistication. Because it has much of the expressive power of LISP the language used for most research in artificial intelligence, LOGO is also valuable for developing large complex programs in graduate level computer science work.

Until recently, most classroom work involving substantial student use of programming languages has been at the research level and has employed minicomputers and larger systems. With the major exception of BASIC, educational programming languages were not even available on microcomputers. During the past year a version of SMALLTALK was implemented on the Z-80

microprocessor, and several implementations of LOGO were completed on microcomputer systems including the TERAK, TI 99/4 and APPLE II. Another LOGO implementation is underway for the Z-80. Thus these contemporary languages will shortly be available on a wide spectrum of educational computer facilities.

The use of a programming language as an important part of student work in a school subject requires the development of a substantial body of curriculum material, for teachers as well as students. In a subject such as mathematics it is possible to develop the entire school math program around the use of programming concepts and work. Programming can serve not only as a laboratory tool to replace paper and pencil work on textbook problems, but more fundamentally, as a conceptual framework for developing the major concepts. Illustrative segments of curriculum materials, with associated presentational strategies, have been developed in preliminary research work in various subjects. LOGO teaching materials of this kind have been developed for several areas of elementary and high school mathematics, as well as for various topics in physics, language arts, and music.

Most of the materials thus far developed are incomplete, both in scope of coverage and elaboration of detail. The introduction of microcomputers into schools on a significant scale, and the availability of languages such as LOGO and

SMALLTALK on these systems, will provide the practical impetus for comprehensive development of extended curriculum material.

7. PROSPECTS AND PITFALLS

In this concluding chapter we shall venture out on the thin ice of attempting to characterize the potential of the microcomputer for improving primary education over the next several years. We will also try to identify certain obstacles and pitfalls that must be avoided if the advent of this new technology is to have its full desired effect.

7.1 Changing Expectations

As computers continue to drop in price and become more widely used in schools, it is to be expected that both teachers and students will become much more sophisticated both in their use of computers and in their expectations of what these new devices can, or ought to be able to do for them. It has been the experience of the computer industry throughout its short history that the rising expectations of its customers have consistently outpaced its ability to satisfy them. There is every reason to believe that this pattern will manifest itself with the educational community as well.

The increasing sophistication of the educational users of computer technology will provide a challenge as well as an opportunity. Educational programs that do no more than "turn the pages" will no longer be acceptable, high resolution color

graphic displays and the use of speech will become the rule rather than the exception, and computer-generated coaching will come to be accepted as a reasonable component of most CAI programs. But programs that fully exploit the capabilities of the technology will prove to be hard to write, and even harder to evaluate. The need for careful evaluation of computer-based educational materials will become widely recognized, and the evolution of such materials will be greatly affected by the manner in which this evaluation is performed.

The impact of improved courseware on education is difficult to predict in detail, but it seems evident that it will be substantial. Will it improve basic thinking skills in the way its proponents hope? Will it make excellent teaching tools available to segments of the population whose education is at present below standard? Or will it, on the contrary, still further exacerbate the differences in educational level between the "haves" and the "have nots"? These questions are difficult to answer with any confidence, but they are important, and the answers to them will to some extent be affected by policy decisions that will be made in the next few years.

7.2 Potential Problems

We turn our attention first to certain problem areas that can be identified clearly as posing obstacles to achieving the

full potential of computer-assisted education. The first, and in some sense most obvious, of these obstacles is the generally poor level of educational material available for microcomputers at present. As Melmed (1980) has pointed out, little exists at the present time by way of a readily available inventory of educational software of superior quality. The development of such software is essential before the educational opportunity represented by the declining costs of hardware can be fully realized.

The difficulty of writing good educational software typically is greatly underestimated by newcomers to the field. Just as good textbooks are quite rare, good educational computer programs are hard to find. Bad ones abound. This poses a serious problem for the future of educational computing. As long as the current situation persists, it will remain overwhelmingly likely that most new users of computers in schools will be sadly disappointed in their apparent limitations. Thus, upgrading of courseware, as well as some sort of informal standard setting, such as exists in the textbook field, are requirements if the new technological opportunity is to be realized.

There are a number of ways in which current software can fail to be effective, and it is perhaps useful to describe the more common of these pitfalls, in what we intend to be in a spirit of constructive criticism. The most common problem, as

determined by the survey undertaken for this paper, is to use the computer simply as an automated page turner. Many of the elementary arithmetic programs, for example, are simply performing the function of a pack of flash cards -- presenting problems and answers with no attempt at feedback, other than perhaps a running count of right and wrong answers. Similarly, many of the so-called CAI programs currently on the market simply present the student with a set number of pages of text, and relegate the computer to performing the function of advancing page by page each time the "RETURN" button is pressed. This is even worse than a book, since it gives the student no control whatever over sequencing -- for example, it is usually impossible to go back over a page that one has already seen, without starting the whole program over again.

Interestingly, many of the available programs seem to make the opposite mistake of trying to do too much with the computer. The foreign language program that displays "Comment allez-vous?" on the screen and will accept "How are you", but not "How do you do?", or any other of the numerous idiomatic translations of this phrase, is the result of one such over-ambitious project. The ability of even very large machines and sophisticated programs to deal effectively with natural language is severely limited, and this is especially true of microcomputers. The beginning French student, however, is unaware of, and properly indifferent to,

this problem, and simply becomes frustrated or confused when confronted with it in this context.

One of the most difficult things to achieve in writing any program is to make it "friendly" enough to be used without difficulty by anyone other than the programmer himself, and it is here that much of the available courseware fails to measure up to what is in principle achievable. Programs are written with little or no attention paid to how, and by whom, they are to be used. Thus, very little flexibility is built into them, no prompting is available to help the beginning user find his way, little or no effort is made to provide "graceful" ways to exit from the program, or from subprogram elements back to an upper level from which other subprograms may be entered. Commands may be ambiguous or counterintuitive, and frequently differ from one program to the next. These issues, which appear peripheral to the educational value of the courseware, may in fact be central to it if they impede the easy interaction with the machine that makes CAI attractive and gives it its unique promise and potential.

A primary reason for the generally poor quality of currently available courseware is cost, rather than inherent technological or psychological factors. The sharp contrast, for example, between the many genuinely intriguing and well designed computer-based games and the scarcity of equivalent quality in educational

materials bears eloquent witness to the fact that market forces have created an imbalance between quality and social utility. It remains to be seen whether this trend will continue, or whether the increasing availability of an important new educational resource will spark renewed efforts to exploit it to the fullest.

But the importance of improving educational software must not detract from our appreciation of the equal importance of preparing teachers to use it effectively. This is an area that has been almost totally ignored until now, but there are signs of an awakening interest in it. A few universities now offer relatively extensive programs in computer-related subjects, and the use of computers in education is a more and more frequent topic at the meetings of many professional societies.

What do we mean by adequate teacher preparation? There are those who insist that nothing less than preparation at the level, say, of a master's degree in computer science will suffice. On the other hand, some schools consider that they adequately cover the subject by offering an introductory course in BASIC programming. We favor a position somewhere between these extremes.

It is certainly clear that to use computers effectively, teachers should know enough about them to feel comfortable using them, to be able to deal with the minor, routine problems that

occur in using computer equipment and software, and to make straightforward modifications to standard programs to fit their own needs. More importantly, they should know something about the special aspects of the student-computer interaction, and ideally be in a position to exploit it. From the standpoint of the educator, the computer is just one more in the growing library of technological aids that are at his or her disposal. The approach to teacher preparation for using computers should be in this light, and should emphasize those aspects of the new device that make it useful and unique in the classroom. This may or may not involve making the teacher into an expert programmer, but it certainly implies a familiarity with the uses and abuses of computers in the classroom.

A fact that educators and educational administrators will have to face increasingly in the future is that many of their students will know a lot about computers and how to use them, as a result of activities outside the school. Probably most children in the United States today have had some exposure to electronic games that make use of digital logic. Increasing numbers of youngsters from more affluent families are using and programming microcomputers in their homes. The percentage of school-aged children who will acquire some degree of computer literacy as a result of such extra curricular contacts is increasing and certainly will continue to do so. The more

creative students will find ways to use microcomputers to advantage even for classes in which their use is not required. In some instances this may create some problems; e.g., inequities arising from the fact that not all students have equal access to such tools, and a discomfort on the part of some teachers with the idea that some students are able to use to advantage a tool they themselves do not understand. On the other hand, the desire to keep up with their students may motivate teachers, who might otherwise not do so, to become skilled computer users themselves, and this could have the ultimate effect of facilitating the integration of computer-based activities into the regular curriculum.

A potential problem that may come to haunt the school system that decides to install one or more computers, is that of unanticipated maintenance costs. Inasmuch as this problem is quite different for computers, at least at present, than it is for other pieces of capital equipment, such as typewriters, a brief explanation is in order.

The experienced administrator or purchasing officer knows what he is getting into when he buys a typewriter, a movie projector, or a pencil sharpener. Such things require periodic inspection and maintenance, and eventually become "used up" and have to be discarded. These hidden costs are usually well understood and are budgeted in a straightforward fashion.

Computers are different, however, in that in addition to the usual maintenance and depreciation costs, which may be higher than they would be for more standard equipment, they are also subject to "software maintenance" costs.

The term "software maintenance" is, to the uninitiated, almost a contradiction in terms. Software refers to programs, and once a program has been written and debugged it presumably represent the solution to a problem, and as such should last forever, or at least as long as the machine it was meant to run on does not change its characteristics. But the fact is, as any experienced computer user will attest, programs are almost like living things -- they continually grow and change under the impulse of the changing needs and demands of the user. The program originally intended to teach addition is modified to include multiplication. The program designed to play "hangman" is used to teach the proper ordering of sentences in a paragraph. The operating system of the computer is "updated" to provide more flexible disk handling, and programs originally written for the old system require modification to run on the new. All of these changes represent substantial continuing costs of the computer system, costs that often exceed the original purchase price.

7.3 Educational Promise

Having described some of obstacles that may prevent computers from achieving their full potential for education, we turn now to a description of what we think that potential may be. There are three distinct ways in which computers may be important for education. First, the study of computers themselves is becoming a necessary aspect of education for the modern world. Second, we believe, as we have stated above, that the study of programming as a mental discipline can have important beneficial consequences for cognitive development. Finally, we foresee an increasing integration of computers into other subjects, with important consequences for education generally. A computer is a very powerful, general purpose tool. People who know how to use one effectively can do things that people who do not possess that special knowledge cannot do. This fact has implications for education that have not been widely recognized. The school system that has good access to a broad range of computing facilities and has a high level of computer literacy among its students and faculty has the potential for approaching many learning tasks in a qualitatively different way than does the system that lacks either the access or the computer literacy or both. An analogy can be drawn between the computer and certain other tools that have had similar, though perhaps less dramatic, educational implications in the past.

The growing importance of computers and information technology is an inescapable feature of modern life. It will be tragic if our educational institutions do not rise to the challenge of educating youth to become familiar with the promises as well as the problems presented by this new and powerful technology. Ironically, it is the adults who often seem to have the most trouble adapting to changing times. Where computers have been made available to them, children have typically accepted and learned to use them more readily than the adults who must teach about them. In light of these facts, we believe it is evident that early instruction in and familiarity with computers are justified on practical as well as intellectual grounds. We anticipate that in time "computer science" will become as basic a subject as reading, writing and arithmetic are today.

It seems at least possible, if not probable, that the wide availability of microcomputers will change fairly drastically some of the traditional roles and functions of teachers and classrooms. To the extent that some of the things that are currently taught by teachers can be taught as well or better by suitably programmed computers, the time that teachers had devoted to these things will be available to be used in more productive ways. The availability of effective instructional resources outside the classroom should help destroy the stereotyped view of the classroom as <u>the</u> place where one goes to learn. The fact

that instruction, at least in some subjects, will not be paced by the teacher's availability and teaching schedule will permit some students to advance at an accelerated pace in subjects of particular interest to them.

We have dealt with the subject of programming as a cognitive skill in section 3.4. To recapitulate, we feel that experiments have demonstrated the unique benefits of interactive programming in teaching formal and heuristic thinking methods and problem solving skills. The utility of programming for other subjects, notably the sciences, is also a valid reason for teaching it at a relatively early level. It is important to emphasize here the difference between programming and mathematics. In the latter subject there is a "right" and a "wrong" answer. The child is taught to adapt to a binary and finite world, in which he finishes a problem, is told that it is either right ("perfect") or wrong ("worthless"), and in either case goes on to something else. Programs are not like that. They are almost never right the first time -- not even when the teacher writes them. They have to be debugged -- a process well worth learning and that extends to other intellectual tasks. Furthermore, programs are like living things. They are always being modified, improved, "customized" to the demands and taste of the particular user. There is no "right" and "wrong" -- just varying degrees of utility, elegance, or beauty. This approximates much more

closely the tasks that the child will face in the "real world" outside of school.

Halfway through the 15th century, the invention of movable type destroyed forever the virtual monopoly on learning of a small elite, and made the world of books accessible to the general population. It made possible an era of mass education the effects of which the world is still exploring. But in making education routinely available to large numbers of people, the printing press took away much of the spontaneity and interactiveness of the Socratic tutoring method which it supplanted. We believe that the advent of the inexpensive computer has the potential for affecting education in significant ways which we can only dimly perceive. If it can help to restore the interactive nature of good teaching without reverting to the elitism of an earlier time, its effects will be revolutionary in the best sense of the word.

References

Abelson, H. MIT LOGO for the Apple II. LOGO Project, Massachusetts Institute of Technology, February 1981.

Alderman, D.L. Evaluation of the TICCIT computer-assisted instruction system in the community college. Educational Testing Service, Princeton, N.J., 1978.

Bates, M., & Wilson, K. ILIAD: Interactive language instruction assistance for the deaf. BBN Report No. 4771. Bolt Beranek and Newman, Cambridge, MA, Sept. 1980.

Bjork, L.-E. An introductory computer programming course and some of its effects on the teaching of mathematics. In O. Lecarme & R. Lewis (Eds.), Proceedings IFIP 2nd World Conference "Computers in Education". Amsterdam: North Holland, 1975.

Brown, J.S., & Burton, R.R. Diagnostic models for procedural bugs in basic mathematical skills. Cognitive Science, 1978, 2, 155-192.

Brown, J.S., & Burton, R.R. SOPHIE - a pragmatic use of artificial intelligence in CAI, 1974.

Carbonell, J.R. AI in CAI: An artificial intelligence approach to computer-assisted instruction. IEEE Transactions on Man-Machine Systems, 11(4), 1970.

Chambers, J.A., & Sprecher, J.W. Computer assisted instruction: current trends and critical issues. Communications, Association for Computing Machinery, 23(6), June 1980.

Clancey, W.J. Transfer of rule-based expertise through a tutorial dialogue. Report STAN-CS-79-769. Dept. of Computer Science, Stanford University, Sept. 1979.

Dresden Associates. School Microware, a directory of educational software. Dresden, Maine, 1981.

DuBoulay, B. Teacher transformation: Student teachers programming in LOGO. DAI Paper No. 122, Dept. of Artificial Intelligence, University of Edinburgh, 1978.

Dugdale, S., & Kibbey, D. The fractions curriculum, PLATO elementary school mathematics project. Computer-based Education Research Laboratory, University of Illinois, Urbana, Ill., March 1975.

Edwards, J., Norton, S., Taylor, S., VanDusseldorp, R., & Weiss, M. Is CAI effective? Assoc. Educational Data Systems, Summer 1974, 7, p. 4.

Feurzeig, W. Computer systems for teaching complex concepts. BBN Report No. 1742, Bolt Beranek and Newman Inc., March 1969.

Feurzeig, W., Goldenberg, E.P., Mattlage, R., & Stachel, R. Demonstration of mixed-initiative CAI capabilities in SSA training applications. BBN Report No. 4043, Bolt Beranek and Newman Inc., June 1979.

Feurzeig, W., Lukas, G., & Lukas, J. The LOGO programming language - learning mathematics through programming. Entelek, Newbury, Massachusetts, 1977.

Feurzeig, W., Lukas, G., Faflick, P., Grant, R., Lukas, J.D., Morgan, C.R., Weiner, W.B., & Wexelblat, P.M. Programming languages as a conceptual framework for teaching mathematics. BBN Report 2165, Bolt Beranek and Newman, Inc., Cambridge, MA, June 1971.

Feurzeig, W., Papert, S., Bloom, M., Grant, R., & Solomon, C. Programming language as a conceptual framework for teaching mathematics. BBN Report No. 1889, Bolt Beranek and Newman, Inc., November 1969.

Gable, A., & Page, C.V. The use of artificial intelligence techniques in computer-assisted instruction: an overview. Int. J. Man-Machine Studies, 12, 1980.

Goldberg, A. Computer assisted instruction: The application of theorem proving to adaptive response analysis. Tech. Report 203, Stanford University Institute for Mathematical Studies in the Social Sciences, 1973.

Howe, J.A.M., O'Shea, T., & Plane, F. Teaching mathematics through LOGO programming: an evaluation study. Dept. of Artificial Intelligence, University of Edinburgh, 1979.

Kemeny, J.G., & Kurtz, T.E. BASIC programming, Second Edition. New York: John Wiley and Sons, Inc., 1971.

Levien, R.E. The emerging technology: Instructional uses of the computer in higher education. New York: McGraw-Hill, 1972.

Malone, T.W. What makes things fun to learn? A study of intrinsically motivating computer games. Xerox Palo Alto Research Center, Palo Alto, CA, August 1980.

Melmed, A.S. Information technology for the schools and nation. Unpublished manuscript, 1980.

Milner, S. The effects of computer programming on performance in mathematics. Proc. Annual Meeting of the American Educational Research Association, February 1973.

Moshowitz, A. On approaches to the study of social issues in computing. Communications, Association for Computing Machinery, March 1981, 24(3).

Murphy, R.T., & Appel, L.R. Evaluation of the PLATO IV computer-based education system in the community college. Educational Testing Service, Princeton, N.J., 1977.

National Center for Educational Statistics: Fast Response Survey System. Student use of computers in schools. U.S. Department of Education, March 1981.

Nickerson, R.S., Perkins, D.N., & Smith E.E. Teaching Thinking. Appendix A in Project Intelligence: The Development of Procedures to Enhance Thinking Skills, Final Report, Phase I. Submitted to Ministry for the Development of Human Intelligence, Republic of Venezuela May 15, 1980.

Olds, H.F., Schwartz, J.L., & Willie, N.A. People and computers: Who teaches whom? Newton, MA: Education Development Center, Inc., September 1980.

Papert, S. Teaching children thinking. Programmed Learning and Educational Technology, 1972, 9.

Papert, S., Watt, D., DiSessa, A., & Weir, S. An assessment and documentation of a children's computer laboratory. Final Report, Brookline LOGO Project, Artificial Intelligence Laboratory, Massachusetts Institute of Technology, Sept. 1979.

Purser, R. Software Directory. California: El Dorado, 1981.

Ragsdale, R.G. Evaluating curricula for the teaching of programming; Topics in instructional computing. Special Interest Group on Computer Uses in Education, ACM, December 1978.

Resnick, C.A. Computational models of learners for computer assisted learning. Doctoral dissertation, University of Illinois at Urbana-Champaign, 1975.

Rubin, A. Making stories, making sense. Language Arts, March 1980.

Sharples, M. A computer based language workshop. Bulletin, Special Interest Group on Computer Uses in Education, 14(3), Association for Computing Machinery, July 1980.

Sheingold, K. Issues related to the implementation of computer technology in schools: A cross-sectional study.Children's Electronic Laboratory Memo No. 1, Preliminary report to the National Institute of Education, Conference on Issues Related to the Implementation of Computer Technology in Schools, Washington, D.C., February 19, 1981.

Sugarman, R. A second chance for computer-aided instruction. IEEE Spectrum, August 1978.

Williams, M., Hollan, J., & Stevens A. An overview of STEAMER: An advanced computer-assisted instruction system for propulsion engineering. Behavior Research Methods & Instrumentation, 1981, 13(2), 85-90.

Wing, R.L. Use of technical media for simulating environments to provide individualized instruction. Board of Cooperative Educational Services, Westchester County, N. Y., 1965.

Zucker, A.A. What is a minicomputer in the school used for? A case study. Unpublished manuscript, U.S. Department of Education, Office of Planning and Budget, March 1981.